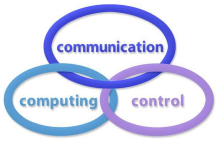
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 18, Issue: 4, Month: August, Year: 2023
Article Number: 4874, https://doi.org/10.15837/ijccc.2023.4.4874



CCC Publications

Deep Learning TCP for Mitigating NLoS Impairments in 5G mmWave

R. Poorzare, A. Calveras Augé

Reza Poorzare*
Department of Network Engineering
Universitat Politècnica de Catalunya, Spain
08034 Barcelona, Spain
*Corresponding author: reza.poorzare@upc.edu
Data-centric Software Systems (DSS) Research Group at the Institute of Applied Research
Karlsruhe University of Applied Sciences
Karlsruhe, 76133, Germany
reza.poorzare@h-ka.de

Anna Calveras Augé
Department of Network Engineering
Universitat Politècnica de Catalunya, Spain
08034 Barcelona, Spain
anna.calveras@upc.edu

Abstract

5G and beyond 5G are revolutionizing cellular and ubiquitous networks with new features and capabilities. The new millimeter-wave frequency band can provide high data rates for the new generations of mobile networks but suffers from NLoS caused by obstacles, which causes packet drops that mislead TCP because the protocol interprets all drops as an indication of network congestion. The principal flaw of TCP in such networks is that the root for packet drops is not distinguishable for TCP, and the protocol takes it for granted that all losses are due to congestion. This paper presents a new TCP based on deep learning that can outperform other common TCPs in terms of throughput, RTT, and congestion window fluctuation. The primary contribution of deep learning is providing the ability to distinguish various conditions in the network. The simulation results revealed that the proposed protocol could outperform conventional TCPs such as Cubic, NewReno, Highspeed, and BBR.

Keywords: Deep learning, 5G, millimeter-wave, TCP.

1 Introduction

Reliable end-to-end communication that guarantees the delivery of individual packets mostly relies on TCP (Transmission Control Protocol) [1], which is one of the inseparable parts of the Internet. TCP employs an acknowledging system to assure that all packets are reliably delivered. Nevertheless,

deploying mmWave (millimeter-wave) in new cellular generations such as 5G (Fifth Generation) for dispensing high data rates makes it difficult for TCP to function sufficiently. The reason is that most materials can act as obstacles between a UE (User Equipment) and a gNB (gNodeB), so reduce the transmission channel bandwidth. Frequent transitions from LoS (Line-of-Sight) to NLoS (Non-Line of Sight) states impair the performance of TCP as the protocol cannot distinguish whether drops are due to congestion or ones forced by NLoS states. The reason for blockage underutilization is that when the channel bandwidth is degraded during NLoS states, some losses caused by buffer overflows can be forced [2], [3]. This issue is more severe in urban deployments as numerous obstacles and mobile users create countless changes in users' states [4]. As a result, most of the conventional TCPs, such as HighSpeed [5] , Cubic [6] , BBR (Bottleneck Bandwidth and Round-trip propagation time) [7] , and NewReno [8], cannot be adapted to the new generations of mobile networks and need some modification to utilize the full potential of 5G mmWave networks [9].

Among the protocols, HighSpeed can reach high performance due to its aggressive congestion control mechanism. On the other hand, BBR, a model-based protocol that strives to achieve high performances through low latencies, is not very successful in fulfilling its aspiration in 5G mmWave networks as the declined bandwidth due to the blockage can mislead the protocol in estimating the bottleneck bandwidth. The other two protocols show intense deficiencies, especially NewReno, whose AIMD (Additive Increase Multiplicative Decrease) has been the based approach in designing other protocols [10] A thorough analysis of TCP over 5G mmWave networks can be found in [2].

Some new protocols have tried to solve this problem by proposing new congestion control algorithms, such as FB-TCP (Fuzzy-Based TCP) [4], DL-TCP (Deep-Learning TCP) [11], [12] for disastrous situations, or even a decision tree algorithm to enhance the congestion control on 5G IoT (Internet of Things) networks [13], and NexGen D-TCP [14]. The first protocol could function appropriately in urban deployments by achieving higher throughput than the other TCPs through low latencies. DL-TCP could reach high throughput for UAVs (Unmanned Aerial Vehicles) in disastrous situations; however, it lacks a well-designed evaluation method as the protocol was trained and tested in the same topology. Furthermore, it has not been compared to HighSpeed, one of the best candidates for the 5G mmWave networks. The frailty of NexGen D-TCP is similar to DL-TCP as it has not also been compared to HighSpeed. Moreover, the feasibility of machine learning approaches in predicting the congestion status in 5G mmWave has been investigated in [15]. The first step was learning the congestion status by deploying some parameters that are important for the transport layer, such as delay, then analyzing different machine learning algorithms. The outcomes indicated that unsupervised schemes are powerful tools for detecting congestion in the network. Furthermore, the proposal of a hybrid deep-learning strategy in [16] for combining LSTM (long short-term memory) and SVM (Support Vector Machine) could show enhancements in the congestion control mechanism by reaching an accuracy of more than 93 percent. All these machine-learning techniques and the obtained results are indicators that intelligent congestion control algorithms can be of assistance in order to hone the functionality of the 5G mmWave network.

Some other machine learning efforts focused on other layers than the transport layer, like deploying an intelligent approach for having a well-designed handover mechanism to tackle the blockage issue of 5G mmWave networks, such as [17], which strive to hone the conditional handover.

Furthermore, there have been some efforts to use MPTCP (Multipath TCP) [18] as an enabler in the exploitation of more than one NIC (Network Interface Card) simultaneously, such as in [19], [20], which strived to mitigate the mmWave impairments by aggregating the network with LTE; however, this mechanism also needs some modifications to be adapted to the 5G mmWave network [21], [22].

The most significant reason for TCP's deficiency is that the protocol is not able to differentiate LoS and NLoS states from each other. If this feature can be embedded in the congestion control mechanism of TCP, it can enhance the performance of the protocol in 5G mmWave networks. For filling this gap, in this paper, we propose a new protocol called DB-TCP (Deep learning-Based TCP) established on the DNNs (Deep Neural Networks) [23] to enhance the functionality of TCP over 5G networks in urban deployments. The new congestion control mechanism of DB-TCP is able to determine different states of the network before changing the sending rate and adjust the cwnd (congestion window) based on the condition that the network is and novel parameters. To the best of our knowledge, this is the

first work to create a new protocol for 5G urban deployments based on DNNs. Extensive simulations revealed that DB-TCP could perform better than conventional TCPs regarding different KPIs (Key Performance Indicators).

The rest of the paper is as follows. Section II presents the necessary backgrounds for deep learning and then proposes the DB-TCP. Section III incorporates the simulation results, and finally, section IV concludes the paper.

2 Deep learning-based TCP

2.1 Introduction to Deep Learning

Deep learning is one of the state-of-the-art techniques employed to solve complicated problems. A DNN is a network established using neurons as inputs, outputs, and hidden layers, i.e., the neurons between inputs and outputs. The network mechanism is to solve problems through forward and backward propagation mechanisms with the help of the neurons and calculations done in these neurons. DNNs are the evolved and improved version of ANNs (Artificial Neural Networks), so they can employ more layers to achieve higher accuracy. The neurons in individual layers receive the previous layer's output, then, using nonlinear functions, i.e., activation functions, calculate the new values and feed them to the next layer; this procedure is called forward propagation. Then based on a reversed approach called backward propagation, the network tries to improve its accuracy and attain a well-suited network to the problem. Different nonlinear functions such as Sigmoid, Relu, and Tanh can be deployed based on various factors such as the training set [23].

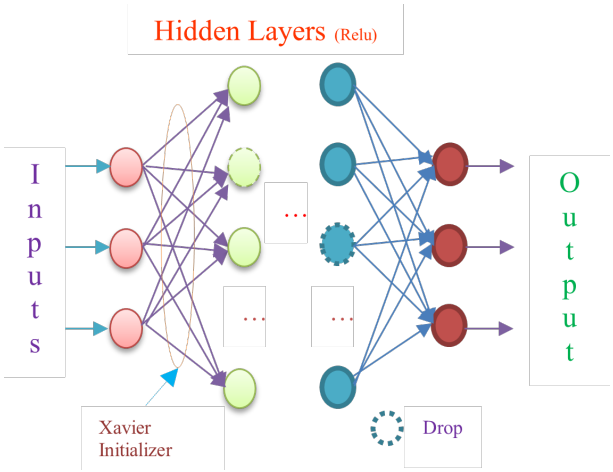


Figure 1: Deep Neural Network Architecture for DB-TCP

Figure 1 shows a DNN architecture and employed parameters in our training. The details of its architecture are discussed in the following subsection.

2.2 DB-TCP Architecture

DB-TCP employs a network of five inputs as the features, three hidden layers, and three outputs. When there is only one hidden layer, the DNN will be capable of handling linear functions and decision-making problems. As the number increases to two, it can solve mapping problems from finite inputs to finite outputs. However, including three hidden layers means that the DNN can solve arbitrary and complicated issues; as a result, we have decided to set the number of hidden layers to three. Regarding the number of neurons in the hidden layers, we have deployed orders of the features along with increasing and decreasing numbers so that the activation function can accomplish higher accuracy. Moreover, Dropout can benefit from this set of numbers.

The employed activation function in the DNN is Relu (Rectified Linear Activation Function), one of the popular activation functions. Based on its mechanism, this activation is one of the most used functions. To prevent overfitting, Dropout [24] has been used, and in order to establish the initial

weights, Xavier Initializer [25] has been chosen. For the calculation of the backward propagation to improve the model, the Adam optimizer [26] has been exploited to lower the loss function and enhance the accuracy. Eventually, Softmax was employed at the output layer to classify the network into three different clusters. The first hidden layer consists of twenty neurons, the second one twenty-five, and the third one twenty.

Considering the parameters to be fed to the network, the first input is the current RTT (Round Trip Time), which is the minimum RTT for the current window. The subsequent inputs are CSI (Congestion Status Indicator), CAD (Cwnd Adjuster), Diff (Difference), and Average. CSI is the result of dividing basertt (the minimum RTT value for the connection) by minrtt (current minimum RTT). CSI is calculated by equation (1):

$$1 : CSI = baseRtt/minRtt,$$

CAD can be obtained by dividing targetedcwnd by currentcwnd. The targetedcwnd is the optimal value for cwnd, and a connection can reach the available throughput by being set to this value; in other words, it is the minimum size of the congestion window that provides maximum throughput under the current access link conditions. Equation (2) indicates how targetedcwnd is calculated where Dthroughput is the desired throughput:

$$2 : targetedCwnd = Dthroughput/minRTT,$$

We should notice that Dthroughput can be calculated by multiplication of baseRtt in current cwnd as shown in equation (3):

$$3 : Dthroughput = currentCwnd * baseRtt,$$

By having targetedcwnd, CAD can be estimated by employing equation (4):

$$4 : CAD = targetedCwnd/currentCwnd,$$

Finally, Diff is the difference between currentcwnd and targetedcwnd and is calculated by equation (5):

$$5 : Diff = currentCwnd - targetedCwnd,$$

Eventually, the last input is the average of the most recent three RTTs, i.e., the sum of the minimum RTTs over the previous three windows. The reason for choosing these inputs is that they can provide a clear insight from the network and help the protocol distinguish NLoS from LoS states. As the blockage can directly impact the RTT, CSI as an indicator factor for baseRtt and minRtt could assist the training engine to have an insight from the network. Besides RTT, the blockage has the potential of changing the desired throughput and optimal cwnd; thus, both CAD and Diff are useful parameters for reflecting the status of the network.

Furthermore, the principal drawback of TCP originated in the fact that it is not capable of detecting the current state of the network and behaving in different conditions in the same manner. As a result, By having these five features, DB-TCP can have a prominent view of the network and adjust the sending rate based on the current condition of the network.

The three outputs of the DNN for specifying the ongoing state of the network are LoS, DNLoS (Dynamic NLoS), and SNLoS (Static NLoS). LoS is when there are no obstacles between the user and the antenna. DNLoS and SNLoS refer to the times when some obstacles act as hurdles on the way to establishing a proper connection. In the former one, the user is moving, but in the latter one, it is still.

The inputs for training the model have been taken from the simulation results of the training scenario. In this scenario, five trees and three buildings act as obstacles to impair communication and create NLoS states.

The detailed information for the obstacles is as follows: the height for trees is ten meters, the distance between trees 1.5 meters, the distance between the last tree and the first building five meters, the height of buildings thirty meters, and the width of buildings eight meters. Furthermore, the user's

distance from the antenna is 68 meters. The antenna that has a fifteen-meter high is connected with a delay of 10 ms (one-way propagation delay) to the PGW (Packet Data Network Gateway), and the PGW is 10 ms away from the application server.

Figure 2 shows the training scenario. The simulation time for collecting data was 70 seconds. However, we should notice that in the final step for the protocol to be applied in practice, numerous obstacles should be employed, and the simulation should be run for days in different movement types to have vast and generalized data.

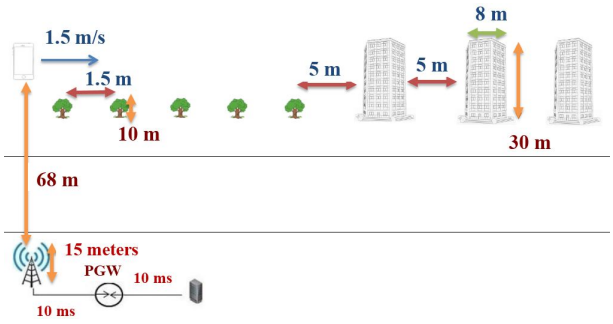


Figure 2: The training topology

DB-TCP can be a starting step for creating novel intelligent TCPs for 5G mmWave, especially over urban deployments. Researchers can use the provided codes and training sets as guidelines for designing new protocols. More importantly, the provided architecture in Tensorflow [27] , the protocol's code in C++, and the way that weights have been included in the protocol for having a connection between Python and C++, are other advantages that researchers can take.

The primary reason for choosing this scenario is that it includes most conditions in urban deployments so that the protocol can function well in cases with fewer flaws.

During training the model, we have used callbacks to stop the training when the accuracy is above 0.9985; as a result, the training has stopped in 987 epochs, and the corresponding loss was 0.0045. Both accuracy and loss are close to the optimal values, which are one for the accuracy and zero for the loss.

Any trained model should be tested in an environment where it has not already seen the data. Consequently, we designed another scenario with different circumstances to mimic the evaluation scenario. There are two obstacles in the evaluation scenario located in other places.

The main aim of this procedure is to figure out how the trained engine functions on inputs that it has not seen before. The result for accuracy and loss was 0.9940 and 0.0367, which are desirable for a trained model.

After determining the state of the network by the outputs of the DNN, DB-TCP adjusts the cwnd based on CSI and Diff, where the first one specifies the aggressiveness of the protocol and the second one is for controlling the steps and preventing the protocol from taking blind paces in inflating the cwnd in a way that cwnd adjustment can be set according to the congestion status. These parameters get updated in every acknowledgment to be adapted to the network's condition in proper time stamps.

Table 1 indicates how DB-TCP adjusts the cwnd in every acknowledgment. The first step in the adjustment process is detecting the ongoing state of the network, which is provided by deep learning, and assists in making proper decisions.

The values in Table 1 have been obtained after numerous simulations to be best fitted for urban deployments. However, the protocol is flexible, and the values are tunable for different situations and deployment scenarios. In other words, we have tried to select the best values through numerous simulations, yet there might be conditions that these values can not function properly, as can also happen to other TCPs.

The rules in this table have been embedded in the congestion avoidance phase of TCP. Most TCPs have similar functionalities in all phases except the congestion avoidance one. This phase is the heart of the protocol and is responsible for reacting to different situations and adjusting the sending rate.

Table 1: HOW DB-TCP ADJUSTS THE SENDING RATE

	DB-TCP	cwnd adjustment
1.	(LoS) and (CSI >= 0.99) and (Diff == 0)	cwnd=cwnd + cwnd/10
2.	(LoS) and (CSI >= 0.99) and (Diff > 0) and (Diff <= 2)	cwnd=cwnd + cwnd/100
3.	(LoS) and (CSI >= 0.99) and (Diff > 2)	cwnd=cwnd + cwnd/1000
4.	(LoS) and (CSI < 0.99 and (CSI > 0.8) and (Diff < 2)	cwnd=cwnd + cwnd/200
5.	LoS) and (CSI < 0.99 and (CSI > 0.8) and (Diff > 2)	cwnd=cwnd
6.	(LoS) and (CSI < 0.8) and (Diff < 2)	cwnd=cwnd-cwnd/100
7.	(LoS) and (CSI < 0.8) and (Diff > 2)	cwnd=cwnd-cwnd/50
8.	(DNLoS) and (CSI >= 0.7)	cwnd=cwnd- cwnd/20
9.	(DNLoS) and (CSI < 0.7) and (CSI >= 0.3)	cwnd=cwnd - cwnd/10
10.	(DNLoS) and (CSI < 0.3)	cwnd=cwnd - cwnd/5
11.	(SNLoS) and (CSI >= 0.6)	cwnd=cwnd - cwnd/5
12.	(SNLoS) and (CSI < 0.6)	cwnd=cwnd/2

3 Simulation results

In order to evaluate the presented protocol, we have conducted extensive simulations in both training and evaluation scenarios and compared the results to four common TCPs. We have exploited the 28 GHz spectrum to satisfy the mmWave requirements, and the carrier frequency was 1 GHz to fulfill the high carrier frequency of mmWave communication. Moreover, RLC (Radio Link Control) buffer size has been set to 2.5 MB in order to accommodate the BDP (Bandwidth Delay Product) value of the network, where the overall propagation RTT is 20 ms and the sending is 1000 Gbps. MSS (Maximum Segment Size) and MTU (Maximum Transmission Unit) are 1400 bytes and 1500 bytes, respectively, identical to their default values, and TcpSocket's maximum transmitting and receiving buffer size equals 6400 KB. We have set RTO to one second as its default value. Finally, the simulation time for the training scenario was 70 seconds, and for the evaluation scenario, 15 seconds.

Moreover, the user's speed is 1.5 m/s, and it stops behind each building in the training scenario and stops for three seconds in the evaluation scenario to mimic a real urban scenario. Furthermore, four different BERs (Bit Error Rates), including small (1.25e-10), moderate (1.25e-9), large (1.25e-8), and zero, have been exploited to emulate congestion and random packet drops in a wireless channel. The training data set and all the codes for the training and simulations can be found in [28]. This Git repository contains all the requirements to re-implement the scenario by a third party, including the training and testing data sets, the ns-3 (Network Simulator) mmWave C++ codes for both data set gathering and evaluation scenarios, the Python code for training the machine learning engine, and finally the C++ code for the DB-TCP implementation. Furthermore, one of the most well-known modules called ns3-mmWave has been employed to obtain the results [29].

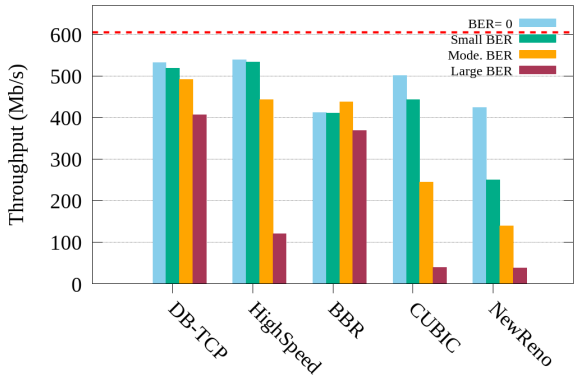


Figure 3: Average throughput comparison for different TCPs in the training scenario

As shown in Figure 3, DB-TCP can operate close to the UDP saturated value, which is 604.99 Mb/s, and has better functionality than other TCPs, especially when there are random packet drops in the network. When there are no packet drops, most TCPs can perform well in these ideal environments.

However, a protocol needs to have a suitable performance when packets drop, especially drops that originated from different causes, such as congestion, channel quality degradation, or blockage. The figure shows that the only protocol that could accomplish the aforementioned goal is DB-TCP. The protocol has stable functionality in all situations, which is desirable for a TCP. The detailed behavior of the other protocols can be found in our previous works [10]. The reason for the sufficient functionality of DB-TCP is that after exiting the slow start phase due to a loss or exceeding the slow-start threshold, TCP remains in the congestion avoidance phase continually, and the proper functionality of the phase in DB-TCP is critical in achieving high performance.

Based on Table 1, CSI is for controlling the aggressiveness of DB-TCP. When the LoS state is ongoing, and CSI is close to one, it indicates that the ideal condition exists and the sending rate can be increased intensely. As CSI starts to decline or Diff gets larger, they are indicators of worsening situations. As a result, DB-TCP slows down in order to drain the buffers and proactively prevent buffer overflows and packet drops. This is to avoid the cwnd from exceeding the available upper bandwidth in the channel. The most important point is that after detecting the current state of the network with the help of DNN. Cwnd is adjusted through CSI and Diff values,

By switching from LoS to NLoS state, CSI becomes smaller and moves toward zero, as the main sign of the blockage is increased RTT. This parameter governs the reduction of the cwnd in DLNoS and SNLoS in order to adapt the sending rate to the available reduced bandwidth because of the blockage problem.

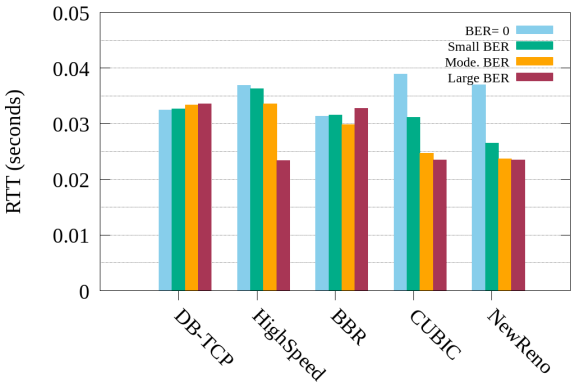


Figure 4: Average RTT comparison for different TCPs in the training scenario

Similar behavior can be drawn from the RTT point of view, as shown in Figure 4. DB-TCP can reduce the RTT value because of its proactive mechanism. It detects the current state of the network and, based on its renewed congestion control mechanism, strives to adapt to the available bandwidth, so the mechanism helps the protocol regulate the in-flight packets in a way that prevents buffer overflows. The fact that DB-TCP can enhance the throughput without harming the RTT (and even improving it) is a significant achievement.

After training DB-TCP and having the different protocols compared in the training scenario, it is time to test the protocol in the evaluation scenario to see how it reacts to the data that it has not seen before. Figure 5 shows the average throughput for various TCPs in the evaluation scenario.

Intriguingly, DB-TCP shows enhancements to other protocols and is the only one that has stable functionality. It is somehow immune to packet drops and functions close to the UDP saturated value, which equals 604.98 Mb/s.

In the lossy environments, which are the adversary environments for TCP, DB-TCP prevents throughput degradation by relying on its innovative behavior, detecting and differentiating LoS, DN-LoS, and SNLoS states, and having an adaptable congestion avoidance phase to 5G mmWave networks.

Figure 6 reveals that DB-TCP could also improve the RTT in the evaluation environment and attain low values compared to other TCPs.

The low RTT for other TCPs in high BERs is because of the low throughput that they achieve. In this case, the transmitted packets are a small number leading to empty buffers. Coming by high throughput through low RTT was also repeated in the evaluation scenario, a considerable achievement

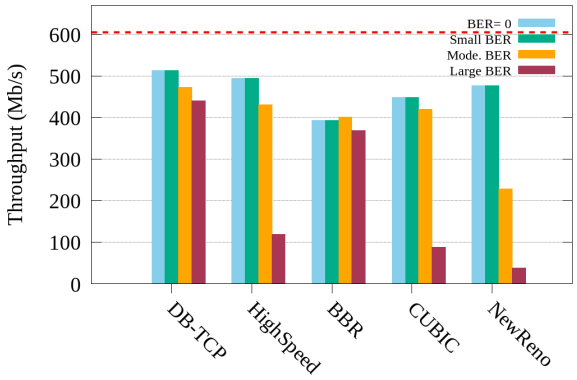


Figure 5: Average throughput comparison for different TCPs in the evaluation scenario

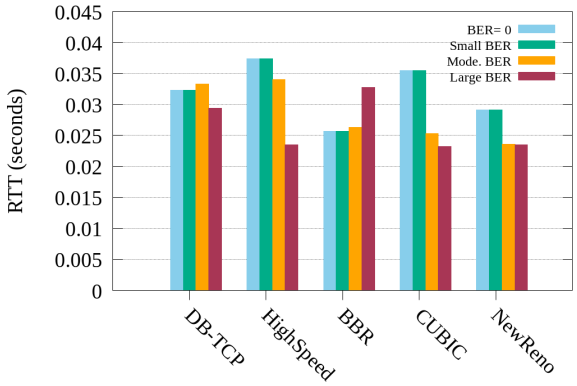


Figure 6: Average RTT comparison for different TCPs in the evaluation scenario

for the protocol. How these protocols behave differently is the question that is going to be answered in the following.

DB-TCP relies on its intelligent congestion control mechanism and detects the network states. Afterwards, it adjusts the sending rate based on CSI and Diff. This mechanism helps the protocol to have stable functionality, respond quickly and accurately to different circumstances, and detect NLoS <-> LoS states.

HighSpeed has an aggressive approach for increasing its cwnd and can recover faster from losses. This mechanism assists the protocol somehow to utilize the high potential of 5G mmWave networks. However, it suffers from two main flaws: 1) by increased packet drop probability, especially non-congested ones, the protocol gets confused in adjusting its sending rate. 2) it blindly increases the cwnd and can exhaust the sender's transmitting buffer based on its congestion avoidance phase rules.

The average cwnd size for DB-TCP when there are no random packet drops in the network equals 1800, however, this value is 27091 for HighSpeed. It is clear that there is a huge gap between these two numbers, and DB-TCP efficiently controls the in-flight packets. The values for other BERs are as follows:

- In the small BER, DB-TCP's cwnd is around 1629, and this value for HighSpeed is 27819, which also is a significant gap.
- In the moderate BER, because of occurring more random packet drops, HighSpeed reduces its sending rate and can function around 14378, in the case that DB-TCP archives 1481.:
- Finally, in the large BERs, HighSpeed backs off dramatically because of the high number of packet drops, and the average cwnd can be 303, which is the main reason behind the protocol's impaired performance. In contrast, DB-TCP adapts the cwnd to the existing conditions by attaining an 1195 average cwnd.

BBR, based on its bandwidth estimation, can have stable functionality, however, misled by degraded bandwidth in blockage states, it cannot function close to the UDP saturated value. The reason is that the degraded bandwidth by blockages misleads BBR in estimating the bottleneck bandwidth accurately. Moreover, in high lossy environments, BBR also gets confused.

The cubic function of CUBIC, which aids the protocol to regulate the cwnd modification based on the time elapsed before the last drop, is the fundamental characteristic of the protocol in moving fast toward the targeted cwnd, i.e., the cwnd size before the last drop. In this case, when the protocol is far from the targetted cwnd, it moves at fast paces. In contrast, it decelerates the cwnd increment speed when converging to its goal. This approach also loses efficiency by appearing congested and non-congested packet drops because by having frequent drops, the elapsed time between losses gets shortened and leads to the CUBIC deficiency.

NewReno, due to its AIMD congestion control mechanism, is the weakest TCP as it cannot adapt itself to high-speed, lossy- environments. The cause is rooted in the congestion control mechanism, which is not for networks that have drops except congested ones.

To sum up, in both scenarios, DB-TCP has better functionality than other TCPs. This superiority is because of its vision of the network's conditions and precisely adjusting the cwnd, as shown in Figure 7. The figure shows that DB-TCP sufficiently reacts to blockage, and when the network enters a blocked situation, the protocol reduces the cwnd. In contrast, after finishing the blocked situation, DB-TCP increases cwnd immediately to the desired value, which is the one accommodated to the bandwidth. Besides this elaborate regulating, the figure reveals that packet drops cannot affect the functionality of DB-TCP in controlling the sending rate, as the protocol shows similar cwnd adjustment in all BERs. Furthermore, the figure indicates that DB-TCP can estimate the lower and upper bound for cwnd and functions between them based on the feedback.

As a result, the most critical tool that DB-TCP employs in controlling the network's status is cwnd and the way it adjusts the sending rate based on the feedback it gets from the network. DB-TCP strives to accommodate the sending rate to the available bandwidth by preventing blind increment or decrement of cwnd, so the average cwnd size achieved in DB-TCP is lower than in other protocols.

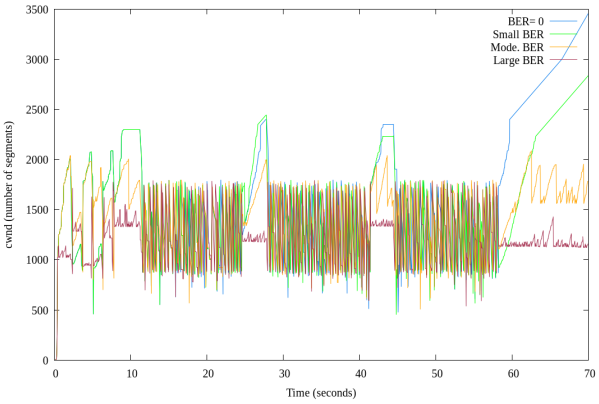


Figure 7: DB-TCP cwnd adjustment in the training scenario

The small average cwnd size has some advantages, such as preventing buffer overflows, avoiding sender buffer exhaustion, functioning around low RTTs, and preventing functionality fluctuations.

For more clarity, to have a detailed look at the instantaneous results, the comparison of the DB-TCP and HighSpeed as the best representative of the other TCPs, has been brought in the following for the training scenario in terms of throughput and RTT.

Figure 8 indicates the throughput comparison for DB-TCP and Highspeed when BER is zero. The figure shows that the cwnd initialization for HighSpeed due to blockage is more than DB-TCP as it encounters no cwnd initialization. This is the main leverage for DB-TCP for attaining higher throughput. It can adjust the sending rate during the blocked state in a way that prevents buffer overflow in the network.

By having a clear vision from the blocked side, DB-TCP can also achieve lower RTT, as denoted in Figure 9. Both protocols have the same functionality in LoS conditions, however, when NLoS is the

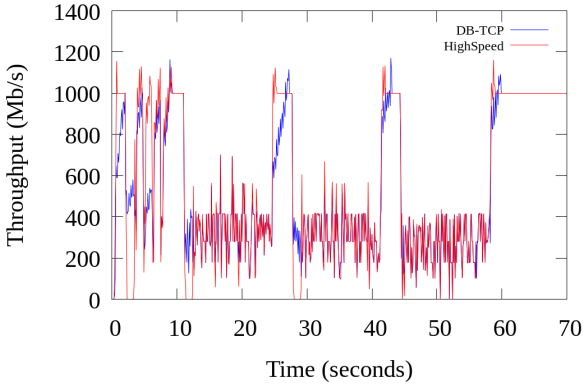


Figure 8: Throughput comparison for DB-TCP and HighSpeed, BER=0

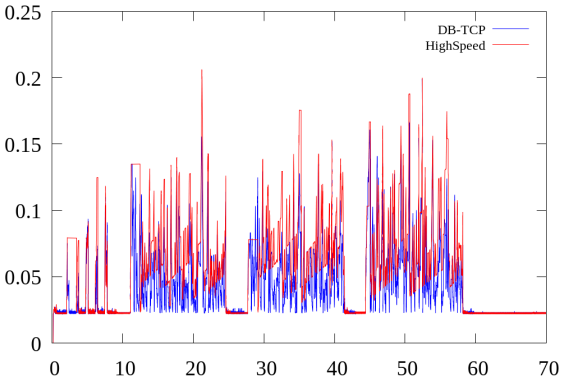


Figure 9: RTT comparison for DB-TCP and HighSpeed, BER=0

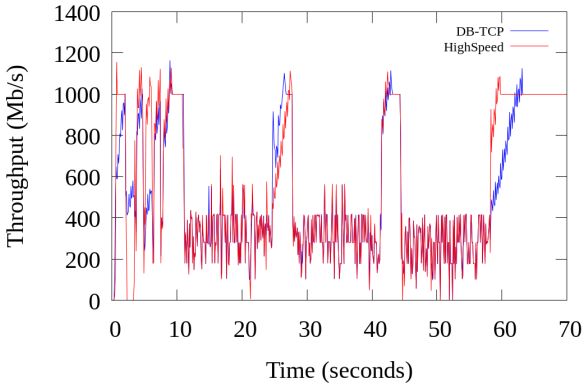


Figure 10: Throughput comparison for DB-TCP and HighSpeed, small BER

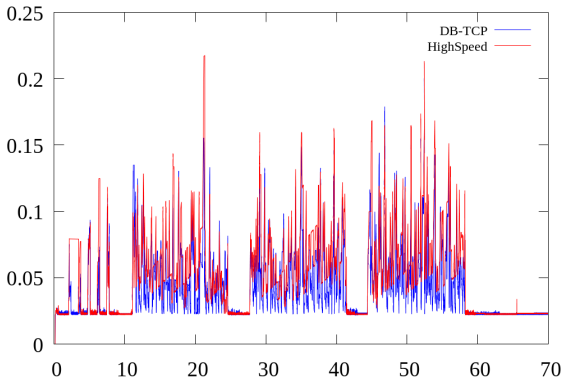


Figure 11: RTT comparison for DB-TCP and HighSpeed, small BER

ongoing condition, DB-TCP can keep the RTT low by sending the appropriate number of transmitted packets into the network. The obtained results for the zero BER were repeated for the small one by a slight difference, as can be seen in Figure 10 and Figure 11

Both figures showed that DB-TCP could keep its superiority compared to HighSpeed when a small number of packet drops exists.

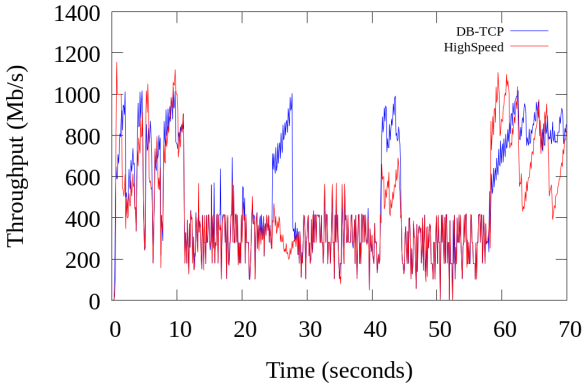


Figure 12: Throughput comparison for DB-TCP and HighSpeed, moderate BER

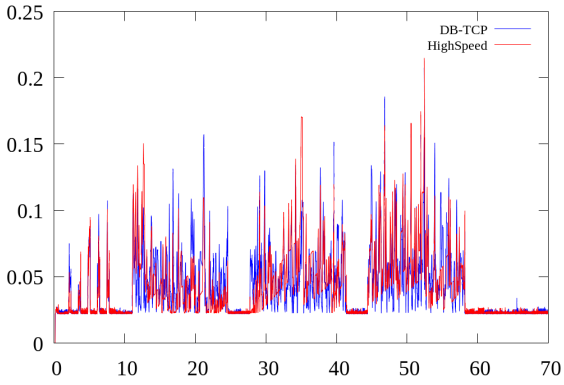


Figure 13: RTT comparison for DB-TCP and HighSpeed, moderate BER

When the BER was increased to its moderate value, HighSpeed could be affected intensely due to its loss-sensitive congestion avoidance phase. In terms of throughput, DB-TCP could outperform HighSpeed most of the time, leaning its clear view from the network's current condition, as seen in Figure 12. The figure shows that both packet drops and blockage mislead HighSpeed so it is tough for the protocol to recover to the highest available performance. In terms of RTT, the value could be

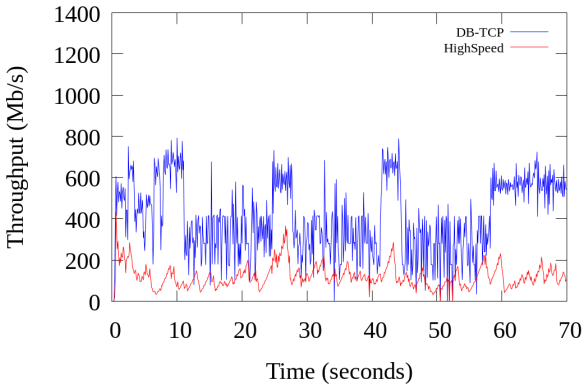


Figure 14: Throughput comparison for DB-TCP and HighSpeed, large BER

reduced for HighSpeed as the number of sent packets declined. Nonetheless, DB-TCP could keep its advantage as the previous BERs, seen in Figure 13.

Finally, when BER was increased to a high value, HighSpeed lost its functionality and could not achieve higher throughput as the high number of losses can mislead the protocol.

As shown in Figure 14, HighSpeed's throughput is always lower than DB-TCP. The high number of packet drops could mislead the DB-TCP performance too, however, it is not very intense.

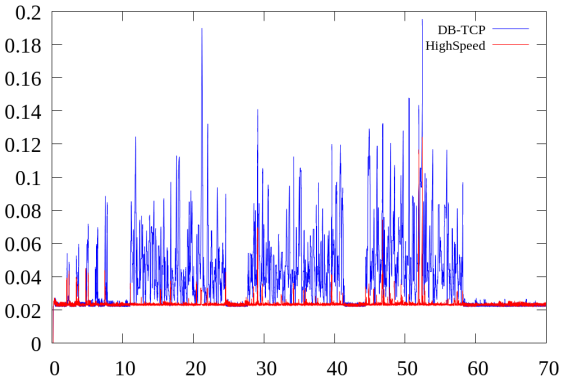


Figure 15: RTT comparison for DB-TCP and HighSpeed, large BER

AS the number of sent packets declined for HighSpeed dramatically, it could be concluded that RTT would also be reduced, as shown in Figure 15. Nevertheless, this low RTT is not an achievement for the protocol considering its accomplished throughput.

To sum up, looking at the instantaneous throughput and RTT of DB-TCP and HighSpeed revealed the supremacy of the latter protocol. DB-TCP owes this attainment to the intelligent view that the protocol has from the network, and leveraging this feature, it is able to distinguish LoS states from NLoS ones.

4 FB-TCP or DB-TCP, which one is the best choice

In this paper, we have introduced DB-TCP, which is necessary to be compared to our previous protocol called FB-TCP [4]. Now it is time to get to the conclusion of which of the protocols has superiorities compared to the other one. For this, we have deployed FB-TCP in both training and evaluation scenarios of DB-TCP, i.e., scenario one and scenario two, respectively.

As it is shown in Figure 16, both protocols have close functionalities, especially when BER is increasing. Looking at the low BERs reveals that DB-TCP has a better performance compared to FB-TCP. This is because this protocol can understand the network well based on its deep learning approach and reacts to different situations precisely.

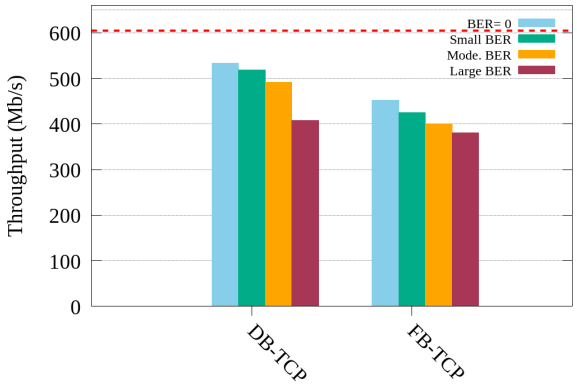


Figure 16: Average throughputs for DB-TCP and FB-TCP in scenario one

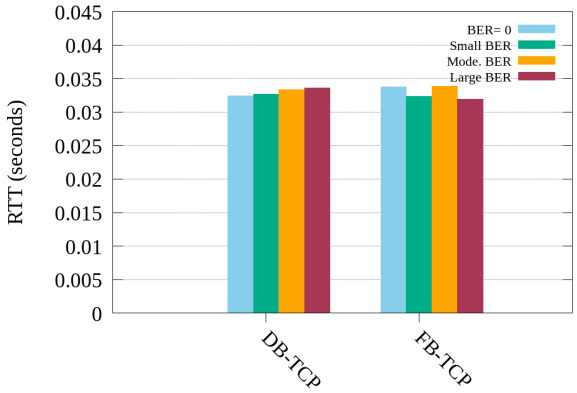


Figure 17: Average RTTs for DB-TCP and FB-TCP in scenario one

This superiority can also be gained in terms of RTT, as seen in Figure 17. Except for the lossy environment, in which FB-TCP has a better RTT, DB-TCP can have negligible improvements in other circumstances. However, in the case of RTT, both protocols can function close to each other.

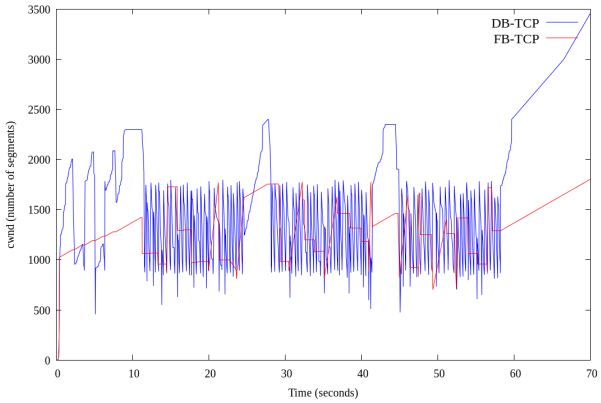


Figure 18: cwnd adjustment comparison of DB-TCP and FB-TCP, BER=0

Comparing cwnd adjustment for both protocols can give us a clear understanding of how they react to various situations. As a result, we have analyzed their behavior in all BERs. Figure 18 shows the cwnd adjustment for both protocols with no random packet drop in the network.

In this case, FB-TCP can have stable functionality all the time; however, DB-TCP reacts properly and recovers quickly from adverse situations, which is its principal superiority. These quick reactions can be seen when NLoS states are finished. Each NLoS state happens when the UE is behind an obstacle. These states are more clear when the UE is behind a big obstacle like a building, as we have

three of them in the figure causing degradation on the performance of the protocols. The impacts of the trees can be seen at the beginning of the figure too.

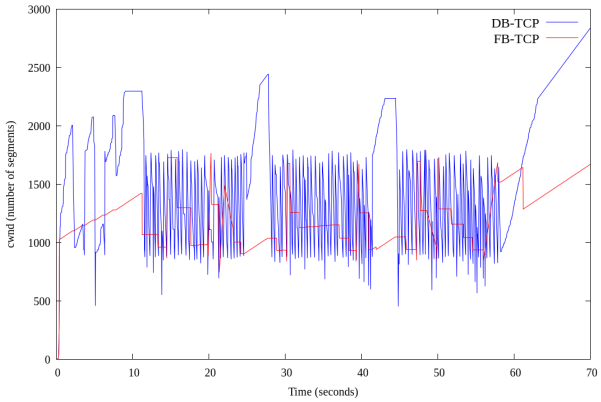


Figure 19: cwnd adjustment comparison of DB-TCP and FB-TCP, small BER

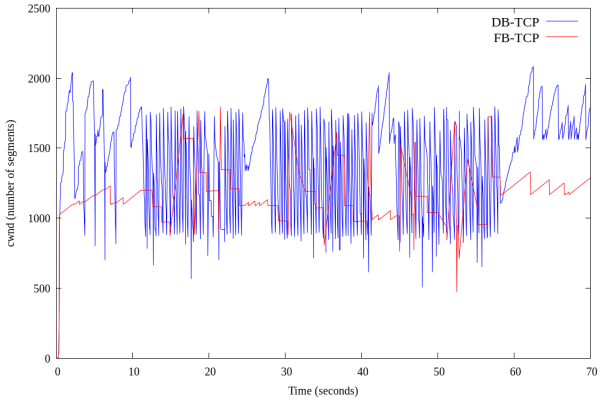


Figure 20: cwnd adjustment comparison of DB-TCP and FB-TCP, moderate BER

Looking at Figure 19 and Figure 20 indicate that small and moderate BERs cannot affect the functionality of protocols because of their non-loss-based nature. However, by having moderate random packet drops, cwnd adjustments are affected minorly, and because of that, both protocols experience a paltry reduction in their performances. Figure 21 justifies this claim and shows that even large BERs cannot mislead these protocols in adjusting the cwnd in contrast to conventional TCPs.

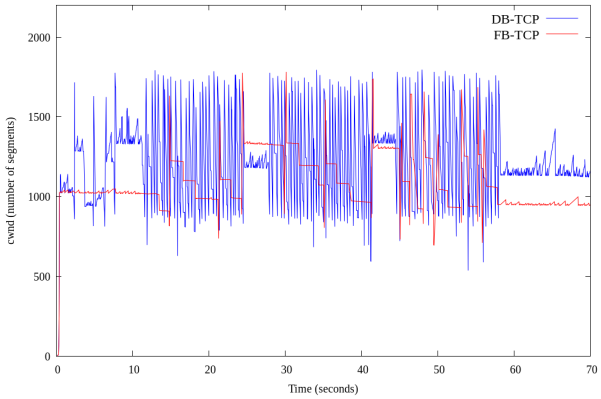


Figure 21: cwnd adjustment comparison of DB-TCP and FB-TCP, large BER

Long NLoS states can affect the functionality of the protocols severely. However, DB-TCP and FB-TCP can distinguish different situations more clearly and reveal the adverse impacts. The reason

Table 2: DB-TCP AVERAGE CWND SIZE COMPARED TO FB-TCP

BER	DB-TCP	FB-TCP
zero	1800	1401
Small	1629	1264
Moderate	1481	1144
High	1195	1061

Table 3: AVERAGE RTTS FOR DB-TCP AND FB-TCP IN SECONDS

BER	DB-TCP	FB-TCP
zero	0.032422	0.033753
Small	0.032685	0.032342
Moderate	0.033330	0.033807
High	0.033607	0.031941

behind this superiority is that these protocols rely on their intelligent congestion control mechanism and try to avoid blind decisions that unwanted situations may force.

For more clarity, we can compare average cwnd sizes for these protocols as seen in Table 2. Both protocols can achieve high performances through low values, which is a significant upside in preventing buffer exhaustion. DB-TCP can attain higher averages cwnd, which is its key capability in having larger throughputs than FB-TCP.

From the RTT point of view, both protocols have the same functionality with some slight changes in various situations, as seen in Table 3.

After comparing the protocols in the first scenario, i.e., training, we have employed FB-TCP in the second scenario, i.e., evaluation, to have in-depth information from both protocols' functionalities. Table 4 summarizes the obtained results for the most important KPIs. In all BERs, DB-TCP has higher throughputs than FB-TCP. However, FB-TCP, because of its reduced throughputs compared to DB-TCP, can achieve lower RTTs. Furthermore, both protocols can function around small average cwnd sizes, which is a positive feature.

To sum up, both DB-TCP and FB-TCP can function adequately in urban deployments by having a clear view of the network's different conditions, such as LoS, NLoS, or random packet drops. Both can achieve high throughputs; however, DB-TCP always has the higher ones. In terms of RTT, the protocols can achieve acceptable RTTs; nonetheless, the lower RTTs for FB-TCP in some cases can be compensated for higher throughputs of DB-TCP. Finally, Both protocols can perform by attaining a small average cwnd size, which can prevent bufferbloating. In a nutshell, it is true that both protocols are excellent choices, but DB-TCP is the most suitable protocol for urban deployments that can satisfy all the expected features of a well-performed protocol. The planned future work is to extend the prototype by collecting more data to have a generally trained engine that can be implemented in real scenarios over urban deployments to ease the way on the path of practical usage of the protocol.

To sum up, As the conventional congestion control mechanisms have no clue about the network's current condition, they cannot make decisions that are suitable for different situations. The most misleading factor in the 5G mmWave network is the transitions between LoS and NLoS states that

Table 4: DB-TCP AND FB-TCP AVERAGE VALUES COMPARISON IN THE SECOND SCENARIO

BER	DB-TCP	FB-TCP
zero	Throughput: 512/76 Mb/s RTT: 0.032340 s cwnd: 1466	Throughput: 450/67 Mb/s RTT: 0.028273 s cwnd: 1150
Small	Throughput: 512/76 Mb/s RTT: 0.032340 s cwnd: 1466	Throughput: 450/67 Mb/s RTT: 0.028273 s cwnd: 1150
Moderate	Throughput: 473.10 Mb/s RTT: 0.033321 s cwnd: 1481	Throughput:444/13 Mb/s RTT: 0.028788 s cwnd: 1133
High	Throughput: 439.76 Mb/s RTT: 0.029359 s cwnd: 1136	Throughput: 402/25 Mb/s RTT: 0.027870 cwnd: 993

impair the protocols' functionality dramatically as they are not able to distinguish packet losses rooted in other issues such as congestion or fading from those caused by NLoS conditions.

The main leverage for DB-TCP is that by deploying a trained machine-learning engine capable of differentiating LoS states from NLoS ones, it can control the sending rate more intelligently. This mechanism helps the protocol make decisions based on the current status of the network and solves the blind decision-making procedure of its predecessors.

5 Conclusion

A new 5G mmWave protocol called DB-TCP was proposed in this paper to overcome flaws caused by NLoS states in urban deployments. The novel protocol relies on deep learning to have a manifest insight from the network's states and adjust the sending rate accurately. As a result, it can achieve higher performance than other TCP variants in terms of throughput, RTT, and cwnd fluctuation. In non-lossy environments, the throughput enhancement can be negligible. However, in lossy ones, it can reach large orders. The main reason for this superiority is that DB-TCP has a tangible view of the network and can react quickly to different states.

Funding

This research was funded in part by the Spanish MCIN/AEI/ 10.13039/501100011033 through project PID2019-106808RA-I00", and by Secretaria d'Universitats i Recerca del departament d'Empresa i Coneixement de la Generalitat de Catalunya with the grant number 2021 SGR 00330.

Acknowledgement

The paper reflects several results obtained in a Ph.D. study carried out at the Universitat Politècnica de Catalunya. Departament d'Enginyeria Telemàtica and contains parts of the doctoral thesis entitled "Contribution to reliable end-to-end communication over 5G networks using advanced techniques" posted on the website of the university (https://upcommons.upc.edu/handle/2117/373401).

Author contributions

The authors contributed equally to this work.

Conflict of interest

The authors declare no Conflicts of interests/Competing interests.

References

[1] Postel, J (1981); Transmission Control Protocol *RFC 793*, Updated by: RFC 1122, RFC 3168, RFC 6093, RFC 6528 [Online]. Available: https://tools.ietf.org/html/rfc793

[2] Poorzare, R; Calveras Augé, A (2020); Challenges on the Way of Implementing TCP Over 5G Networks, *IEEE Access*, 8, 176393 - 176415, 2020.

[3] Zhang, M; et al (2019); Will TCP Work in mmWave 5G Cellular Networks?, *IEEE Communications Magazine*, 57(1), 65-71, 2019.

[4] Poorzare, R; Calveras Augé, A (2021); FB-TCP: A 5G mmWave Friendly TCP for Urban Deployments, *IEEE Access*, 9, 82812-82832, 2021.

[5] Floyd, S (2003); HighSpeed TCP for Large Congestion Windows *RFC 3649*, [Online]. https://tools.ietf.org/html/rfc3649

[6] Ha, S; Rhee, I; Xu, L (2008); CUBIC: a new TCP-friendly high-speed TCP variant, *SIGOPS Operating Systems Review*, 42(5), 64–74, 2008.

[7] Cardwell, N; Cheng, Y; Gunn, C. S; Yeganeh, S. H; Jacobson, V (2016); BBR: Congestion-Based Congestion Control, *Queue*, 14(5), 20–53, 2016.

[8] Henderson, T; Floyd, S; Gurtov, A; Nishida, Y (2012); The NewReno Modification to TCP's Fast Recovery Algorithm, *RFC 6582*, [Online]. https://tools.ietf.org/html/rfc6582.

[9] Hindawi, B; Abbas, A. S (2021); Congestion Control Techniques in 5G mm Wave Networks: A review, *2021 1st Babylon International Conference on Information Technology and Science (BICITS), Babil, Iraq*, 305-310, 2021.

[10] Poorzare, R; Calveras Augé, A (2021); How Sufficient is TCP When Deployed in 5G mmWave Networks Over the Urban Deployment?, *IEEE Access*, 9, 36342-36355, 2021.

[11] Na, W; Bae, B; Cho, s; Kim, N (2019); DL-TCP: Deep Learning-Based Transmission Control Protocol for Disaster 5G mmWave Networks, *IEEE Access*, 7, 145134-145144, 2019.

[12] Kuppusamy, S.P; Subramaniam, M; Gunasekar, T (2023); Deep learning-based TCP congestion control algorithm for disaster 5G environment, *Preprint*, 2023.

[13] Najm, I.A; Hamoud, A.K; Lloret, J. S; Bosch, I (2019); Machine learning prediction approach to enhance congestion control in 5G IoT environment, *Electronics*, 8(6), 607, 2019.

[14] Kanagarathinam, M.R; et al. (2020); NexGen D-TCP: Next Generation Dynamic TCP Congestion Control Algorithm, *IEEE Access*, 8, 164482-164496, 2020.

[15] Diez, L; Fernández, A; Khan, M; Zaki, Y; Agüero, R (2020); Can We Exploit Machine Learning to Predict Congestion over mmWave 5G Channels?, *Applied Sciences*, 10(18), 6164, 2020.

[16] Khan, S; Hussain, A; Nazir, S; Khan, F; Oad, A; Alshehri, M.D (2022); Efficient and reliable hybrid deep learning-enabled model for congestion control in 5G/6G networks, *Computer Communications*, 182, 31-40, 2022.

[17] Lee, C; Cho, H; Song, S; Chung, J-M (2020); Prediction-Based Conditional Handover for 5G mm-Wave Networks: A Deep-Learning Approach, *IEEE Vehicular Technology Magazine*, 15(1), 54-62, 2020.

[18] Ford, A; Raiciu, C; Handley, M; Bonaventure, O; Paasch,C (2020); TCP Extensions for Multipath Operation with Multiple Addresse *RFC 8684*, [Online]. https://datatracker.ietf.org/doc/html/rfc8684.

[19] Polese, P; Jana, R; Zorzi, M (2017); TCP and MP-TCP in 5G mmWave Networks, *AIEEE Internet Computing*, 21(5), 12-19, 2017.

[20] Polese, P; Jana, R; Zorzi, M (2017); TCP in 5G mmWave networks: Link level retransmissions and MP-TCP, *in Proc. IEEE Conf. Comput. Com- mun. Workshops (INFOCOM WKSHPS), Atlanta, GA, USA*, 343-348, 2017.

[21] Poorzare, R; Waldhorst, O.P (2023); Toward the Implementation of MPTCP Over mmWave 5G and Beyond: Analysis, Challenges, and Solutions, *IEEE Access*, 11, 19534-19566, 2023.

[22] Mahmud, I; Lubna, T; Cho, Y-Z (2022); Performance Evaluation of MPTCP on Simultaneous Use of 5G and 4G Networks, *Sensors*, 22(19), 856-858, 2022.

[23] Schmidhuber, J (2015); Deep learning in neural networks: An overview, *Neural Networks*, 61, 85-117„ 2015

[24] Srivastava, I; Hinton, G; Krizhevsky, A; Sutskever, I; Salakhutdinov, R (2014); Dropout: A simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, 15(1), 1929-1958, 2014.

[25] Glorot, X; Bengio, Y (2010); Understanding the difficulty of training deep feedforward neural networks, *in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 9, 249-256, 2010.

[26] Kingma, D.P; Ba, J (2014); UAdam: A method for stochastic optimization, *International Conference on Learning Representations (ICLR)*, 1412-6980, 2014.

[27] Google; Google. TensorFlow, Accessed: May. 2021. [Online]. Available: https://www.tensorflow.org

[28] Poorzare, R; DB-TCP's Source Codes, Accessed: Apr. 2021. [Online]. Available: https://github.com/rezapoorzare1/DB-TCP/tree/main.

[29] Mezzavilla, M; et al. (2018); End-to-End Simulation of 5G mmWave Networks, *IEEE Communications Surveys and Tutorials*, 20(3), 2237-2263, 2018..

Cite this paper as:

Poorzare, R.; Calveras Augé, A. (2023). Deep Learning TCP for Mitigating NLoS Impairments in 5G mmWave, *International Journal of Computers Communications & Control*, 18(4), 4874, 2023. https://doi.org/10.15837/ijccc.2023.4.4874