# Performance Evaluation of Application Aware Multi-TCP Transport Layer Congestion Algorithms in mmWave Network

Dvija Patel*, Jayesh Jakkani*, Harsha Varma*, Abhishek Gupta*,
*Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27606
(dpatel26, jjakkan, suppala, agupta38)@ncsu.edu

## I. Abstract

Transmission control protocol has grown over 50 years to become the dominant protocol on the internet. But the main aim for developing the protocol was to find a reliable protocol which can be implemented on wired network. Since the channel in wired networks is almost constant, TCP did not face any problems, especially with channel variability. Under wireless network, TCP fails to give its optimum performance because it does not take channel behaviour into consideration. So, a need for a proactive TCP protocol is called for. We use NS3 to simulate different TCP protocols and observe their behaviour under mmWave network. After observing the behaviour, we came to a sense that a given particular TCP protocol cannot satisfy the requirement of all the applications. Some applications require high bandwidth whereas few require low latency and rest of the other require low packet loss. In this paper, we propose a TCP framework which will select a particular TCP to satisfy the application demands.

## II. Introduction

Cellular networks have exponentially grown within the past decades and their technologies are constantly advancing. There is a never-ending demand from mobile users for highly reliable connectivity. However, cellular network channels are known for their highly variable channel capacities and fluctuating network conditions. Variations in location and orientation between the user's device and access points, and existence of obstacles cause increased variability in the channel quality. With 5G developing in cellular networks, these impediments will continue to cultivate and interfere with network connectivity and reliability.

Prior research has been devoted to investigating issues presented on the physical and MAC layer. However, some prior research has shown that propagation conditions are more impactful on the transport layer, especially the congestion control mechanisms. The transport layer consists of a TCP protocol that ensures reliable, ordered, and error free delivery of bytes from one application to another.

Some of the known variants of TCP are TCP Tahoe, TCP NewReno, TCP Cubic, and TCP Vegas. However, none of these protocols are well-suited for cellular network conditions where the underlying channel changes at short time scales, nor can they distinguish stochastic packet loss from losses caused by congestion. Advanced protocols like TCP Verus and YeAH congestion protocol aim at adjusting the window size by estimating the channel conditions. However, the protocols channel estimation mechanism may not be too accurate to distinguish temporary signal outages caused by the NLoS occurrences. This is a very common problem that is highly emphasized in mm-Wave cellular networks. Because the transport layer is not notified of the short-term changes, it is unable to instantly decrease the congestion window. This results in immediate action by the lower layers by lowering the Modulation and Coding Scheme to increase robustness against unfavorable channel conditions. So the real question is how can TCP be improved to reduce the above mentioned obstacles in 5G mm-Wave mobile networks?

To produce a more valuable TCP protocol that solves the Latency vs Throughput trade-off in transport protocol in mmWave networks, this research paper considers the following approaches. First, making TCP aware of channel conditions or track the mmWave link state when the channel between the UE(User Equipment ) and the eNB switches from an LOS(Line of Sight) to NLOS(Non Line of Sight) condition. Second, the reason for sub-optimal performance of TCP in mmWave networks is the momentary loss of mmWave link is misunderstood as congestion in network and prompts TCP to dramatically reduce congestion window and only gradually increase the window size thereafter [32]. Our approach is to devise new congestion control algorithm which could take this undesired behavior into consideration and alleviate the problem. Finally we can use Multi-path TCP (MTCP) [22] for multi-connectivity which manages different congestion control algorithm for each sub flow in coupled (dependent) or uncoupled (independently) design. Our modified TCP will amalgamate above mentioned paradigms to see if new Transport protocol can achieve substantial throughput gain with minimal latency as compared to other proposed TCP variations

[19], [20] in mmWave networks which is future of wireless communication. Our proposition is to implement modified TCP - mmWTCP in NS-3 and compare the corresponding results.

Application-aware congestion control protocol: There is a demand for a single transport layer protocol which can fulfil the requirements of all the applications. But it is very difficult to develop an algorithm which can provide both high throughput and low delay. There is always a trade-off between throughput and delay which means the protocol is only able to fulfil one specific application requirement. Every existing protocol satisfies at least one of the requirements that an application demands. So, a framework can be implemented which can select an appropriate protocol depending on the application demands. Sometime, an application can require both high throughput and low delay but this requirement can be needed during particular instance. This specify demand can be fulfilled by switching the appropriate protocol on the fly without disturbing the flow.

## III. Related Work

**WTCP A Reliable Transport Protocol for Wireless Wide Area Network** A very low and variable bandwidth, very high and variable delays, significant non-congestion related loss, asymmetric uplink and downlink channels, and occasional backouts are the characteristics of wireless wide-area networks (WWAN). Wireless TCP (WTCP), a reliable transport protocol is designed to perform efficiently and fairly over WWAN networks. Instead of using window-based transmission control, WTCP uses rate-based transmission control to tackle channel unpredictability. Particularly, WTCP uses inter-packet delay to determine rate adaptation and computes the rate adaptation at the receiver side by varying the granularity of rate i.e. increase or decrease depending on the type of congestion observed. In addition, WTCP perform well for short-lived flows by modifying its startup behavior. The protocol successfully overcomes the problems of inaccurate round-trip time computations, handles asymmetric channels and does not burst packets. However. The protocol does not address the problem of fairness, deep buffers and short-lived flows. [29]

**TCP Westwood Bandwidth Estimation for Enhanced Transport over Wireless Links** TCP Westwood (TCPW) is a congestion control protocol for wired/wireless networks. The protocol makes modifications of the window congestion control scheme on the sender side of a network to cope with the channel variability in wireless networks. TCPW uses end-to-end estimation by controlling the window size which allows it to be totally transparent to routers and to the destination and in turn making it compatible with any network and TCP implementation. The mechanism continuously monitors the ACK reception rate at the sender side by estimating the packet rate of the link. The predicted link rate is then used to evaluate congestion window size and slow start threshold. Continuously resetting the window size by estimating the link rate make the protocol more robust to channel variability. Results from the paper shows that the protocol performances way better than TCP Reno and SACK in mixed wired/wireless networks over high-speed links. However, TCPW performs poorly when random packet loss rate exceeds a few percent that infers TCPW is breaks completely during high error rates. [16]

**Performance Evaluation of Snoop Protocol for Wireless Networks** To cope with congestion, packet losses due to high error rates, signal fading and hand-off problems a new protocol was introduced, Snoop Protocol. The protocol runs as a snoop agent on a base station or on a wireless device. The snoop agent monitors the packet being passed through the base station or the wireless device and caches the packet. After caching, the snoop agent forwards the packet to the destination and waits for the TCP acknowledgement. If the packet is lost or a duplicate acknowledgement is received by the base station or the wireless device from the destination, the snoop agent will retransmit the lost packet locally to the destination without forwarding the duplicate acknowledgement or NACK to the sender. Since the TCP layer is not aware of the packet lost, the congestion control mechanism is not initiated. Adding to this, the snoop agent starts a retransmission timer for each TCP connection. If an acknowledgement is not received for the packet, the snoop agent retransmits the packet and starts another timer called as persist timer. Results of the paper shows that Snoop protocol enhances the performance of TCP over wireless networks. However, it fails to perform over high bit error rates. [18]

**Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks** Most of the recent wireless protocol have low channel utilization which results in low throughput and high delay. Need for a new protocol to address this problem was necessary. Sprout, a wireless end-to-end transport protocol for interactive application that desire high throughput and low delay came into light. The protocol adapts to drastic change in link speeds by observing the packet arrival times which in turn infers about the uncertainty of network path. Monitoring of packet arrival times is used to estimate the number of bytes that can be safely send by the sender without causing any congestion in the network. Evaluation of sprout protocol over cellular network implies that it the best protocol for cellular network when it comes to low delay application. However, for application that require high throughput, the protocol fails extremely as the authors have paced a cap on the throughput knob to achieve high channel utilization. [31]

**On the Use of TCP BBR in Cellular Networks** TCP BBR(Bottleneck Bandwidth and Round-trip propagation time) is a congestion-based algorithm developed by Google. In this research paper it is being compared to TCP Cubic and TCP NewReno, which are the two most commonly used TCP variants. In comparison to older algorithms like CUBIC that depend on loss as a direction towards congestion, BBR approximately measures the feasible bandwidth and minimum round-trip time every certain period. Additionally, NewReno and Cubic use packet loss as indication of congestion. However, loss only occurs when the buffers are close to full at the bottleneck. The congestion is only detected when the

bottleneck is already overloaded, leading to large delays. TCP BBR is considered model-based, since it builds an imitations of the network path between the communicating nodes in terms of bottleneck bandwidth and minimum round-trip delay. It is also considered model based because it attempts to work at the point where all usable bandwidth is consumed and the round-trip delay is at the lowest. Even though TCP BBR outperforms the two, it does not share the available bandwidth in a fair way. [3]

**X-TCP: A Cross Layer Approach for TCP Uplink Flows in mmWave Networks** This paper introduces an alternative TCP congestion control algorithm that uses the information on the SINR and the resource allocation available to the UE in order to tune the TCP congestion window to an optimal value. This allows minimization of the queueing delay in the RLC layer buffers without affecting the throughput in a negative way. The performance of this approach was tested in a randomly generated scenario, with a detailed and realistic end-to-end simulator, and it was shown that the average RTT and RLC buffer occupancy are more than 50% smaller than those of TCP CUBIC. The negative aspect of the proposed approach is a decrease in fairness with respect to legacy flows. [4]

**Adaptive Congestion Control in Cellular Networks** Cellular network channels are known for their highly variable channel capacities and fluctuating network conditions. This results in TCP and its variants to perform inefficiently over cellular networks due to high capacity variability, self-inflicted queuing delays, random packet losses that are not linked to congestion, and large bandwidth-delay products. The Verus protocol is specifically designed for highly variable channel conditions. The key idea of this protocol is to use delay profiles and delay variations to quickly adapt to changing channel conditions. So, instead of attempting to predict the cellular channel dynamics, it uses cues from delay variations to track channel conditions and quickly change its sending window. [32]

**TCP in 5G mm-Wave Networks: Link Level Re-transmissions and MP-TCP** This paper presents the first comprehensive performance evaluation of the interaction of Single Path and Multi-Path TCP with mm-Wave links, with and without lower-layer re-transmission mechanisms, It re-marked that for mm-Wave it is very important to mask the channel losses to the higher TCP layer with link re-transmission mechanisms, otherwise it is not possible to reach high throughput. In addition, it also showed that when the mm-Wave link has a high probability of being in NLOS state, a secondary LTE sub flow improves the throughput performance of long-lived TCP sessions more than a mm-Wave sub flow, and that the design goals of MP-TCP may not be met with mm-Wave links. [24]

**Advanced 5G-TCP: Transport protocol for 5G Mobile Networks** In this paper authors have presented novel high-speed TCP algorithm with help of detail analysis of HSTCP. Increased friendliness and higher data rates are achieved when Advanced 5G-TCP is employed in the predefined simulation environment. This protocol is designed to be used in the future 5G data networks regardless of the engaged MAC Layer. Advanced 5G-TCP represents novel high-speed protocol that uses parabola in order to define the protocol response function and the calculation of decrease factor (w) parameter. If the network capacity allows this protocol could easily reach rates gather than 400Gbps. So, the authors have modified knobs used to control window size to modify existing protocol HSTCP to adapt for 5G mmWave communication. [21]

**Analysis of TCP performance in 5G mm-wave mobile networks** This paper has done a comprehensive analysis, taking into consideration several loss-, delay-based and hybrid congestion protocols and studying their performance for different applications and scenarios. The paper concluded that Congestion control protocols that aim at maximizing throughput through quick recovery like CUBIC suffer with longer (Non-Line of Sight) NLoS periods while they perform comparatively well in the presence of short NLoS. On the other hand, the performance variations of protocols that detect congestion through hybrid mechanisms (packets loss and RTT measurements) like YeAH is minimal. This trade-off is especially evident in presence of handovers. [17]

**A Survey on Recent Advances in Transport Layer Protocols** This paper surveys the main novelties related to transport protocols that have been recently proposed, identifying following three main research trends 1. The evolution of congestion control algorithms, to target optimal performance in challenging scenarios, possibly with the application of machine learning techniques. 2. The proposal of new transport protocols, alternative to the Transmission Control Protocol (TCP) and implemented in the user-space. 3. The introduction of multipath capabilities at the transport layer. [23]

**Transport Layer Performance in 5G mmWave Cellular** The basic question in this paper was to understand how this variability of mmWave channel impacts the end-to-end performance. This evaluation is challenging since the end-to-end performance depends in complex ways on the interactions of the underlying channel dynamics, beam tracking, MAC-layer scheduling and retransmissions, network delays and congestion control. To understand these system-wide implications, authors have studied some representative scenarios using a novel end-to-end ns3-based simulation framework. Paper has revealed that due to the very high data rate, current slow start mechanisms can take several seconds to achieve the full throughput offered by a mmWave PHY-layer. Second, large drops in rate, which are likely to be common in LOS-NLOS transitions, can result in very high levels of queuing and buffering, dramatically increasing the latency. Thirdly, after a retransmission timeout (RTO), even aggressive TCP protocols such as Cubic can take inordinately long to recover to full rates. While RLC and MAC layer retransmissions can shield upper layers from packet losses, RTO will still occur in real situations. [33]

**Performance analysis of RED with different TCP congestion control mechanism** Implementation of active queue management (AQM) in TCP network can minimize congestion to great extent. This paper introduces one of the AQM scheme

i.e. random early detection (RED). The idea behind RED scheme is to drop packets from the buffer when there is a possibility of congestion in the network even though the buffer is not full yet. The scheme is implemented along TCP Reno, Vegas, Sack 1 and Fact. Results show that the TCP variants perform extremely well when implemented along RED but at the same time fails to get the same results when working without RED. However, RED scheme fails to deliver good performance during high packet loses. [13]

**The BLUE active queue management algorithms** Blue algorithm was proposed to overcome the drawbacks faced by some of the AQM schemes. Particularly, the reason behind this drawback was that the AQM schemes used queue lengths as the indicator of the congestion level. BLUE uses completely different knobs to monitor the severity of congestion. BLUE uses packet loss and link idle events to manage congestion. Simulation conducted using BLUE indicated that the scheme outperformed RED, both in terms of buffer size and packet loss rate of the network. However, BLUE AQM scheme does not address the problem of fairness due to which Stochastic Fair Queueing was implemented. [8]

**The Bufferbloat Problem over Intermittent Multi-Gbps mmWave Links** This paper presented the first comprehensive evaluation of bufferbloat on an end-to-end simulation of mmWave cellular links. In this paper simulation methodology employs realistic, detailed measurement-based channel models. These models can capture dynamics in the channel due to the blockage and transitions from LoS to NLoS states, which is the main problem in end-to-end performance. Paper finds that bufferbloat can be severe problem for mmWave cellular systems due to the high variability of the channel combined with the delays in the cellular core network. Moreover, conventional AQM techniques are unable to mitigate the bufferbloat problems. However, we find dynamic receive window can greatly reduce the delay with minimal loss in throughput. Paper present a promising initial result that properly using channel information at the UE can dramatically improve end-to-end performance with relatively simple changes. [34]

**Mitigating Buffer-bloat with Receiver-based TCP Flow Control Mechanism in Cellular Networks** This paper proposes ABRDA (Available Bandwidth based Receiver Window Dynamic Adjustment), a receiver-based solution to mitigate buffer-bloat in cellular networks. ABRWDA obtains the wireless link bandwidth by means of the relationship between bandwidth and SINR, then uses it and the estimated RTT to calculate the receiving window which will be used to adjust the sending window. It compares its performance with the current receiver-based solution and the delay-based TCP version Vegas. The experiment results manifest that ABRWDA can achieve considerable performance benefits compared with other approaches. [15]

### The Role of MPTCP in 5G

MPTCP should be aware of the presence of multiple DC radio links, which may be exposed as a single or distinct network interfaces/IP addresses.

MPTCP should optimally partition traffic or duplicate a data flow over DC links, depending on the application's need, network policy and conditions.

Mobility in 4G networks, as described at the architecture level in [3GPP.23.401], was based on a central mobility solution that made it difficult to relocate mobility anchors closer to the end user. In contrast, 5G uses a distributed mobility solution based on multiple anchors providing different IP addresses as the device moves from one area to another. In 4G networks, session continuity is enabled by anchoring a PDN Connection (as PDU Sessions are referred to in 4G networks) to a P-GW which allocates an IP address to the mobile device: PDN connection and IP address allocation are maintained as long as the device remains attached to the network, even when the device moves around. In 5G, different types of session continuity can be provided, and are indicated by a "Session and Service Continuity" (SSC) mode value of 1, 2 or 3 (defined in [3GPP.23.501] section 5.6.9). Every PDU session is associated with a single SSC mode, which cannot be changed on this PDU session. The following sub-sections will study how 5G handles each SSC mode, and potential effects on MP-TCP.

### MPTCP with 5G Dual Connectivity

One of the key features of 5G [3GPP.23.501] is dual connectivity (DC). With DC, a 5G device can be served by two different base stations. DC may play an essential role in leveraging the benefit of 5G new radio, especially in the evolving architecture with the coexistence of 4G and 5G radios. On a 5G device with DC, an application can send data to the destination (e.g., a single-home server) through multiple radio links, over one or more PDU sessions. Some PDU sessions may be over a single radio link, while others may have flows over each radio link. Therefore, in a first case, DC can be made visible to applications through different IP addresses, while in a second case, DC can be used by different flows terminated at the same IP address on the device. In any of those cases, the issues of out of order delivery and diverse latency values need to be supported in DC. However, such reliable communication scenarios have not been addressed in the current DC architecture. Based on the design history of DC in earlier systems, the 5G system will need to incorporate features to support robustness/reliability (e.g., backup and duplication), that will likely result in added complexity. Meanwhile, similar issues have been solved in MPTCP (e.g., two types of sequences at subflow and connection levels for out of order delivery, etc.). MPTCP hence could be leveraged in those scenarios. On the other hand, to satisfy new QoS requirements of 5G applications as well as to benefit the most from DC, 5G devices are expected to include advanced algorithms that control data transmissions over each radio link optimally. For example, those algorithms could aim to minimize overall latency or satisfy latency requirements of applications. Hence, the 5G device needs to dynamically select the most suitable path for a given radio condition. Additionally, algorithms for shifting, based on congestion, ongoing traffic between transmissions over radio links are also necessary. MPTCP, which includes

path manager, scheduler, and congestion control functions, shows a lot of potential to address the issues mentioned above. MPTCP could, therefore, be integrated with DC and the 5G protocol stack, as an alternative to developing 5G-specific solutions. As part of this integration, the MPTCP stack should be aware of the presence of multiple radio links, whether they are exposed using multiple IP addresses or hidden under a single IP address. MPTCP's scheduler should optimally partition traffic or duplicate a data flow over different links, depending on the application's need, network policy, and conditions.

**Are they all data or can we use some for control? Are the sub-flows independent or coordinated?**

This area is under extensive research. A MPTCP implementation will include path manager, scheduler, and congestion control functions, shows a lot of potential to address the issues mentioned above. MPTCP could, therefore, be integrated with DC and the 5G protocol stack. Refer to the following paper: https://tools.ietf.org/id/draft-defoy-mptcp-considerations-for-5g-01.html As part of this integration, the MPTCP stack should be aware of the presence of multiple radio links, whether they are exposed using multiple IP addresses or hidden under a single IP address. MPTCP's scheduler should optimally partition traffic or duplicate a data flow over different links, depending on the application's need, network policy, and conditions. In one of the research papers [22], the performance of two different congestion control algorithms for MPTCP: BALIA (that is, a coupled CC algorithm), and CUBIC (an uncoupled algorithm) is compared. MP-TCP with the coupled BALIA CC algorithm fails to meet the first target of MP-TCP design, because in many cases its throughput is lower than that of SP-TCP. When coupling a mmWave and an LTE link, indeed, the congestion control algorithm sees the losses on the 28-GHz mmWave connection as congestion, and steers the whole traffic to the LTE subflow, degrading the end-to-end connection's performance. Instead, the uncoupled congestion control algorithm is unaffected by this issue, because each path behaves independently. However, in this case the MP-TCP flow might be unfriendly to SP-TCP flows on shared bottlenecks. Refer to figures 1 and 2.

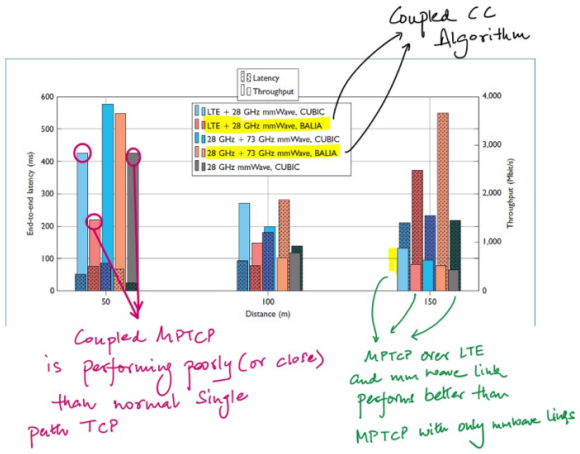


Fig. 1. Multipath TCP (MP-TCP) throughput and latency for different distances d and different MP-TCP options. The rightmost bar in each group shows the performance of a single path TCP (SP-TCP) connection with TCP CUBIC over a 28-GHz mmWave link as a reference. (BALIA stands for Balanced Linked Adaptation algorithm)

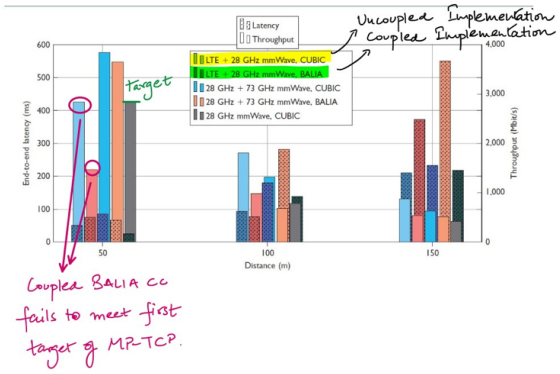


Fig. 2. Multipath TCP (MP-TCP) throughput and latency for different distances d and different MP-TCP options. The rightmost bar in each group shows the performance of a single path TCP (SP-TCP) connection with TCP CUBIC over a 28-GHz mmWave link as a reference. (BALIA stands for Balanced Linked Adaptation algorithm).

**Investigate how 5G hops are different from regular wireless hops. Will current wireless TCP protocols be performing poorly in 5G? Why?**

An optimized interaction between transport protocols and mmWave radio access networks will be a fundamental enabler of 5G networks. TCP and its variants are known to perform poorly over cellular networks due to high capacity variability, self-inflicted queuing delays, stochastic packet losses unrelated to congestion, and large bandwidth-delay products [11]. The lossy nature of mmWave links accentuate this problem of TCP triggering congestion control mechanisms even if there is no congestion [22]. In paper [33] detail evaluation has given what are the problems of traditional TCP on mmWave networks. The basic question in this paper was to understand how this variability of mmWave channel impacts the end-to-end performance. This evaluation is challenging since the end-to-end performance depends in complex ways on the interactions of the underlying channel dynamics, beam tracking, MAC-layer scheduling and retransmissions, network delays and congestion control. To understand these system-wide implications, authors have studied some representative scenarios using a novel end-to-end ns3-based simulation framework. Paper has revealed that due to the very high data rate, current slow start mechanisms can take several seconds to achieve the full throughput offered by a mmWave PHY-layer. Second, large drops in rate, which are likely to be common in LOS-NLOS transitions, can result in very high levels of queuing and buffering, dramatically increasing the latency. Thirdly, after a retransmission timeout (RTO), even aggressive TCP protocols such as Cubic can take inordinately long to recover to full

rates. While RLC and MAC layer retransmissions can shield upper layers from packet losses, RTO will still occur in real situations.

**Should we improve TCP by reporting/using channel information?**

So far, we have got only one research paper [4] which is using channel information to tweak window size in TCP. Two critical behaviours emerge: the bufferbloat issue and the consequent increase in the delay in NLOS conditions, and the slow recovery after extended outages that cause TCP retransmission timeouts. These are the consequences of the abstract view that TCP has of the end to end connection. In order to improve the overall performance paper has introduced an alternative TCP congestion control algorithm that uses the information on the SINR and the resource allocation available to the UE in order to tune the TCP congestion window to an optimal value, that minimizes the queueing delay in the RLC layer buffers without harming the throughput. The performance of this approach was tested in a randomly generated scenario, with a detailed and realistic end-to-end simulator, and showed that the average RTT and RLC buffer occupancy are more than 50% smaller to those of TCP CUBIC. The negative aspect of the proposed approach is a decrease in fairness with respect to legacy flows. The analysis was then extended to a scenario designed to trigger an RTO, and the performance of the cross layer congestion control algorithm was compared to that of TCP BIC, Illinois, CUBIC and NewReno, showing that the proposed approach is the fastest to reach the full utilization of the mmWave link bandwidth.

**Details of MPTCP** [25]

In brief, here is how MPTCP works. MPTCP is negotiated via new TCP options in SYN packets, and the endpoints exchange connection identifiers; these are used later to add new paths—subflows—to an existing connection. Subflows resemble TCP flows on the wire, but they all share a single send and receive buffer at the endpoints. MPTCP uses per subflow sequence numbers to detect losses and drive retransmissions, and connection level sequence numbers to allow reordering at the receiver. Connection-level acknowledgements are used to implement proper flow control.

**How is the connection set up?** The TCP three-way handshake serves to synchronize state between the client and server. In particular, initial sequence numbers are exchanged and acknowledged, and TCP options carried in the SYN and SYN/ACK packets are used to negotiate optional functionality. MPTCP must use this initial handshake to negotiate multipath capability. An MP CAPABLE option is sent in the SYN and echoed in the SYN/ACK if the server understands MPTCP and wishes to enable it. Shown in Figure 3.

**How to add a sub flow?** To open a new subflow, MPTCP performs a new SYN exchange using the additional addresses or ports it wishes to use. Another TCP option, MP JOIN is added to the SYN and SYN/ACKs. This option carries a MAC of the keys from the original subflow; this prevents blind

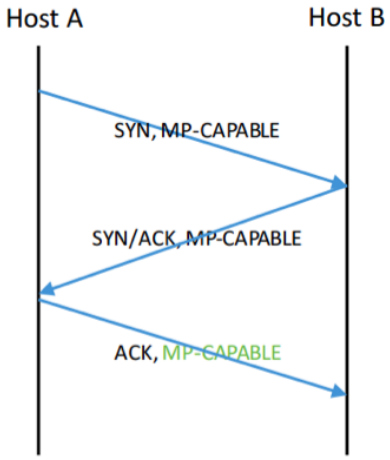


Fig. 3. Connection Setup

spoofing of MP JOIN packets from an adversary who wishes to hijack an existing connection. MP JOIN also contains a connection identifier derived as a hash of the recipient's key ; this is used to match the new subflow to an existing connection. Shown in Figure 4.

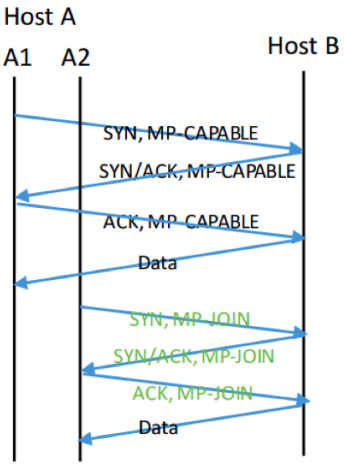


Fig. 4. Sub-flow

**How is the Congestion control handled?** The linux kernel supports MPTCP with LIA(Linked Increase Adaptation). An improved version of this is BALIA(Balanced Linked )

Balia is a generalized MPTCP algorithm that strikes a good balance between friendliness and responsiveness. The algorithm only applies to the AIMD part of the congestion avoidance phase. The other parts such as slow start,

fast re-transmit/recovery algorithms are the same as in TCP [RFC5681]. The minimum ssthresh is set to 1 MSS instead of 2 when more than 1 path is available. Each source has a set of paths r. As a special case, the set can be a singleton in which case Balia reduces to TCP Reno. Each path r maintains a congestion window w_r and measures its round- trip time rtt_r. The window adaptation of Balia is as follows: - For each ACK on path r, increase w_r by:

$$\frac{x_r}{\tau_r\left(\sum x_k\right)^2}\cdot\left(\frac{1+\alpha_r}{2}\right)\cdot\left(\frac{4+\alpha_r}{5}\right)$$

Note that Balia's decrement algorithm multiplies the MD algorithm of TCP Reno by a factor in the range of [1, 1.5].

$$\frac{w_r}{2}\cdot\min\{\alpha_r,\ 1.5\}$$

$x_r := w_r/\tau_r \qquad \alpha_r := \frac{\max\{x_k\}}{x_r} \qquad \tau_r$ is the round trip time

**How is the Flow control handled?** If we inherit TCP's interpretation of receive window, this would imply an MPTCP receiver maintains a pool of buffering per sub-flow, with receive window indicating per-sub-flow buffer occupancy. However this can lead to a deadlock scenario: 1. Assume that the next packet that needs to be passed to the application was sent on sub-flow 1, but was lost. 2. In the meantime sub-flow 2 continues delivering data, and fills its receive window. 3. Sub-flow 1 fails silently. 4. The missing data needs to be re-sent on sub-flow 2,but there is no space left in the receive window, resulting in a deadlock.

This problem is solved by each connection a single receive buffer pool should be shared between all sub-flows. The receive window then indicates the maximum data sequence number that can be sent rather than the maximum sub-flow sequence number. As a packet present on a different sub-flow always occupies the same data sequence space, no such deadlock can occur.

**How is the Connection and sub-flow are torn down?** MPTCP's FIN semantics also allow sub-flows to be closed cleanly while allowing the connection to continue on other sub flows. Finally, MPTCP provides a REMOVE ADDR message, allowing one sub-flow to indicate that other sub-flows using the specified address are closed. This is necessary to cleanly cope with mobility when a host loses the ability to send from an address and so cannot send a sub-flow FIN.

**Should we be using an MP-TCP link setup in which multiple TCP sub-flows are used ?**

Yes, in 5G the abrupt change from LOS to NLOS means there is a potential drop from any of the sub flows but not all necessarily, by a careful coupled congestion control algorithms like Balanced Link addition Adaptation(BALIA), we can ensure at least few of the flows will be active in sending the packets although other flows are abruptly blocked.

**How do we split the flows (half-and-half)? How many sub-flows?**

Although we can employ traditional load balancing algorithms like Round Robin, Least RTT and least congestion window, it would be better if the flows can be split based on a function of RTT and window size of each flow like in BALIA for an optimized performance.

Although there is no hard constraint on the maximum allowed subflows, Hardware constraints like available free ports can give us an upper bound on the number of sub flows, in addition too many subflows can create processing delays. However, there is not enough evidence supporting any particular number of sub flows per connection.

**Mobility Management - Mobile IP [27], [12], [1], [7], [30]**

Mobility management generally involves protocols at various layers. The network layer specifically uses IP mobility management protocols – macro-mobility (Inter-system domain mobility - movement between two network domains) and micro-mobility (Intra-system domain mobility - movement between 2 subnets in the same domain).

Our main issue is that, as the node moves from one place to another, it results in change of network and/or subnet which consequently results in change of IP address. Since all the connections take IP address as a seed thus change in IP address means that all the connections must be re-established which inevitably leads to interruption in on-going applications and services. This issue of varying IP addresses when node is mobile is resolved by Mobile IP. Mobile IP is a powerful protocol that supports Internet mobility, but highly recommended to be used only in MACRO-MOBILITY. It solves the problem of node mobility by redirecting packets for the mobile node to its current location.

So how does Mobile IP work? [27] The protocol introduces seven elements: (i) Mobile node (MN) – a device or a router that can change its point of attachment to the Internet, (ii) Correspondent node (CN) – the partner with which MN communicates, (iii) Home network (HN) – the subnet to which MN belongs, (iv) Foreign network (FN) – the current subnet in which the MN is visiting, (v) Home agent (HA) – provides services for the MN and is located in the HN, (vi) Foreign agent (FA) – provides services to the MN while it visits in the FN, (vii) Care-of-address (CoA) – defines the current location of the MN; all packets sent to the MN are delivered to the CoA.

The paper [27] lists 3 steps Mobile IP follows: 1) Agent discovery – mobile node is able to detect if it has moved to a different subnet 2) Registration – Once the mobile node receives its care of address, it registers it with its home agent

to inform it of its current location 3) Routing and Tunneling – If a correspondent node is sending the packet it is sent to mobile node's home agent. If the packet is coming from a foreign agent then it gets decapsulated by the mobile node's foreign agent. The obvious downside to this protocol is that it has large overhead.

According to [27] the main drawback of Mobile IP is that it is not an efficient mechanism in a highly mobile scenario as it requires an MN to send a location update to the HA whenever it changes its subnet. The signaling cost for location updates and the associated delay may be very high if the distance between the visited network and the home network is large.

**How is mobility handled in 4G/5G cellular networks?**

The hand-off technique that UE use in 3G network is soft hand-off whereas in 4G they use hard handoff. According to https://www.youtube.com/watch?v=c5NBMwj9ZdA, in hard hand-offs connection is made with the new access point before it breaks the established connection made with the current access point. Source eNodeB regularly requests measurement report from UE and depending on the parameters like intra-frequency neighbour, inter-frequency neighbour and inter-RAT neighbouring cells it decides a target eNodeB. If the source eNodeB realises that the UE can be better served by a different eNodeB (target eNodeB), it will send a request for handover. Depending on various factors like congestion or ongoing events the target eNodeB will conduct admission control mechanism and if it finds fit to include a new UE into its system, it approves the request of source eNodeB. After receiving the approval, source eNodeB send a signal to UE saying that it can change its access point and the same is confirmed by UE to the target eNode. After final confirmation from target eNodeB, hand-off is successful.

**Buffered Packets During Mobility**

According to https://www.qualcomm.com/media/documents /files/lte-mobility-enhancements.pdf, during hand-off, the UE's control plane and user plane context are transferred from the source eNB to target eNB. Also, in order to minimize packet loss and provide in-order delivery, the source eNB forwards the UE's down-link and optionally up-link user plane data to the target eNB. This indicates that buffered data is transferred to target eNB during hand off, thus no packet drop.

**Investigate 5G Link layer reliability.[5], [28]**

In 5G networks, Ultra Reliable Low Latency Communication (URLLC) is implemented to provide link layer services. One of them is error handling, this technique is similar to the technique used for 4G but is expected to perform all the process with low latency. Knowing the faulty nature of the channel, the base station buffers the data that is arrived and request for a resource grant to the target UE. Once request is accepted and the data is transmitted to the UE, the UE decodes the data and responds with either a positive or negative acknowledgement based on the success of data decoding. If the UE fails to send an acknowledgement within an allocated time then the BS re-transmits the data. Compare to LTE, URLLC operates in a shorter Transmission Time Interval (TTI) and requires a faster response from the UE to avoid re-transmission. Also, the BS can re-transmit the data if the ACK/NACK is not received due to channel fading. This can increase latency, and hence wastage of resources. To overcome this issue, stronger channel coding and multiple antenna technologies can be considered.

**For 5G, LoS is a problem and the small size of the coverage area is a problem. How should TCP handle the interruptions?** [34]

So the main question is why do we need to implement LoS in 5G cellular networks? This is because 5G results in higher frequency, which means it will produce low wavelength. If there is low wavelength the signal will be able to propagate through obstacles.

A study was conducted to compare the performance of two types of management queues: Drop-tail and CoDel when experiencing human blockage and building blockage.

The main difference is that, with humans, the channel deteriorates slowly and the blockage only lasts for a short period; on the other hand, with buildings, the link capacity drops rapidly and the blocking interval is much longer.

In the human blockage scenario, the User Equipment is walking at 1 m/s, 300 meters away from the base station, while maintaining LoS connectivity, and experiencing 3 human blocking events. With the use of Drop-tail there was a high throughput, high buffer occupancy and high delay. This was because the RLC queue size is large and any dropped packets are recovered by lower layers. CoDel drops packets when it detects high buffering delay. As the queue drastically builds up during the blockage, the sender is notified to shorten the congestion window. By decreasing the congestion window as the blockage increases, the RLC buffered packets begin to decrease. After the blockage is recovered, the congestion window ramps back up again but very slowly.

In the building blockage scenario, the UE is walking at 1 m/s, 150 meters away from the base station. Drop-tail's results were similar to the human blockage scenario. Compared to human blockage, there is a sudden capacity drop with building blockage. Drop-tail does not enable packet drops at the sender, which results in keeping a large congestion window, high buffer occupancy and large delay because the RLC queue continues to grow. With Building blockage, CoDel becomes even more inefficient due to the sudden capacity drop. More packets are dropped to stop the growth of the queue. Since re-transmission is a slow process, it results in an almost zero throughput. Hence, the TCP ramp up time is dramatically effected.

So how should TCP handle these interruptions? We would like to inherit the idea of proactive scheme which monitors channel capacity similar to those in the Verus and Sprout protocol. Specifically monitor channel capacity with the use of signal to noise ratio.

## IV. System Design

For implementation purposes, we started with a basic network. Refer to Figure 5. This network comprises of a remote host, a gateway, a mmWave eNodeB and a mmWave UE. In initial stages, we kept the mmWave User Equipment is stationary. TCP New Reno is used as an underlay transport protocol in this network.

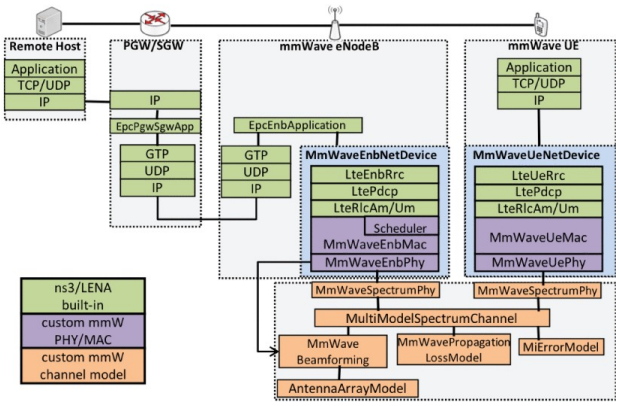


Fig. 5. System Design

Remote Host is an internet server which acts as a database and provide all the file requested by the UE. Gateway provides internet connectivity to the UE. mmWave eNodeB acts as a wireless access point for UE. UE is a mobile device using which users access Internet.

### A. Environment Design

The environment design mainly focuses on the last mile because the system remains the same from remote host to the eNodeB. The only thing that matters in wireless network is the last mile where the channel is different. In this environment, we have an eNodeB, a UE which is mobile and some building models (particularly 3 buildings). Figure 6 depicts the environment design in two directions below;

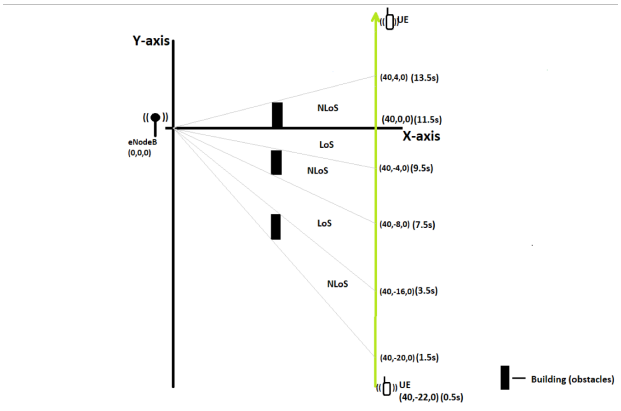


Fig. 6. Environment Design

In our simulation, the UE starts at co-ordinates (40, -22, 0) and is stationary for 0.5 seconds. During this period, there is full connectivity between eNodeB and UE. After 0.5 seconds, the UE starts to moves in Y-axis direction with a velocity of 2 m/s. At 1.5 seconds, the UE is just about to enter in NLoS region and remains till 3.5 seconds (16 meters). Then, the UE enters LoS region at 3.5 seconds and travels up to 8 meters which is equivalent to 7.5 seconds from the time of start. Furthermore, the UE experiences NLoS region two more times i.e. between 7.5 seconds to 9.5 seconds and between 11.5 seconds to 13.5 seconds. The UE is in LoS in the intervals between NLoS and finally at 13.5 seconds the UE receives full connectivity for the rest of simulation.

## V. Methodology

Research over the past few decades has provided us with ample knowledge of various TCP algorithms. To further elaborate, [35] mentions that the wireless links that involve the frequent non-congestion packet loss can be best served by the TCP Veno because of its smart window adjustment technique. On the other hand, Satellite links with very high RTT can use TCP Hybla because of the different Additive Increase Multiplicative Decrease(AIMD) mechanism. Modern day Internet, uses TCP CUBIC and New Reno along with HTCP for High bandwidth and delay. However, the literature suggests using delay-based protocols like TCP Vegas can very well provide low RTTs and can avoid frequent packet drops.

Following is the tabulated explanation of the available TCP congestion algorithms and its advantages.

| Requirement | Network Condition | Congestion Control Algorithm |
|---|---|---|
| Very Long RTT | Satellite Link | TCP Hybla [6] |
| Freq. Non-Congestion Packet Loss | Wireless Link | TCP Veno [9] |
| High Bandwidth, High Delay | Wired/Wireless | TCP Cubic [10] |
| Highly Varying Network Conditions, Buffer Bloating | Wireless Link (Cellular/WiFi) | TCP Vegas [2] |

From the information in the table we can conclude that not all TCPs are created equal and not all applications have the same requirements. Because of this classical problem of many to many matching, we need a robust framework by which we can exploit the advantages of already existing TCP congestion algorithms in serving the applications in the best way possible. In this direction, we implemented a switching engine in the transport layer that could adapt to the TCP congestion algorithms and satisfy the application layer requirements accordingly.

Referencing the above literature, our switching mechanism

chooses a congestion algorithm based on the target parameters(target RTT, target throughput) set by the application, the available throughput and RTT and the underlying network conditions.

We evaluated our framework in an mmWave test bench setup in Network Simulator(NS3) by configuring the target parameters under various channel conditions. The observations will be discussed in the Results section.

## VI. Evaluation

### *A. End to End simulation of mm-Wave network at 1Gbps and 300Mbps*

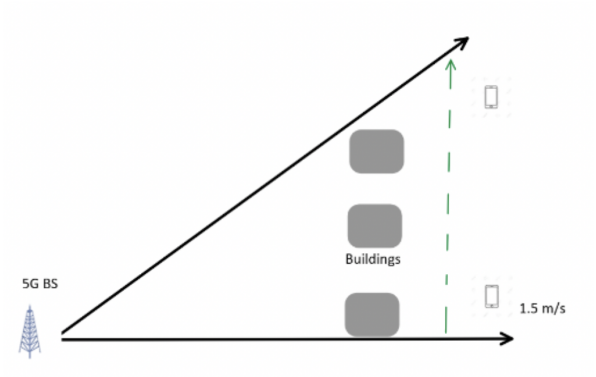


Fig. 7. Test Case Scenario

*1)* ***Test Case Scenario*** *:* Here in Figure 7, we first try to capture LOS and NLOS transition when UE moves in mmWave cell along with randomly places obstacles which are buildings in this case. Using this simulation, we would try to evaluate potential issues of using TCP as it is in mmWave network. We will compare the performance achieved by a TCP flow at varying data rates and RLC buffer sizes. We have used TCP protocol – New Reno which comes as default in Linux. How other congestion control algorithm behave will be studied in detail via a vis RTT, Throughput and Congestion Window Variation. We will give detailed mathematical explanation of how and when UE is moving into LOS (Line of sight) and its effect on above mentioned parameters.

In all the cases because of slow start mechanism, an application might take several seconds to achieve full throughput (Figure 8) offered by mmWave physical layer and thus create issues if they rely on short TCP connections.

Fig 9 depicts how much throughput is achieved when UE is moving from LOS to NLOS and vice-versa. We have done simulations for 1Gbps and 300Mbps data rate. The spikes are observed initially because initial threshold is set very high in NS-3 simulator. On experience congestion (duplicate ACK) threshold is appropriately set and we observe no spikes thereafter.

Buffer- bloat, which causes high latency and packet delay variation when queuing too many packets in buffers, is a

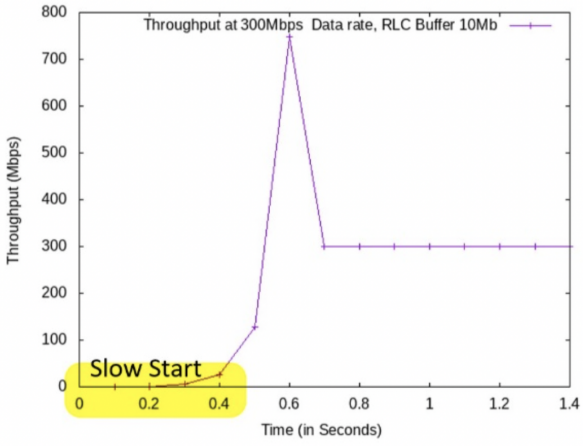


Fig. 8. Zooming the Fig below to Slow Start Phase

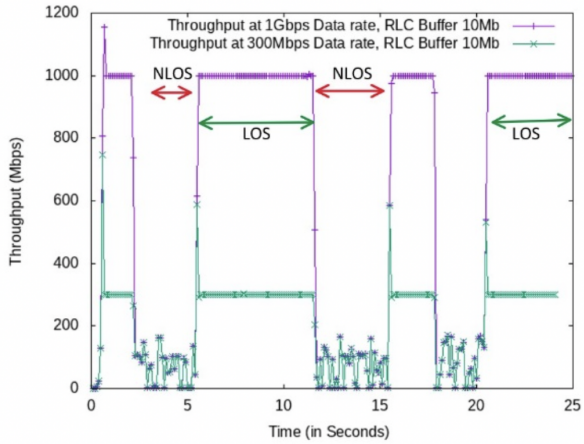


Fig. 9. Throughput at 1Gbps and 300 Mbps

known problem in wireless networks. As shown in Fig 11, with NLOS connectivity the latency dramatically increases.

Due to high capacity of mmWave networks, Congestion window can grow very large. Due to lower layer transmissions on detection of errors, RTO timer does not expire, and packets keeps on accumulating in buffers. This leads to buffer bloat problem and introduces higher latency as depicted in peaks circled red.

If we compare the solid green and dotted blue lines in Fig 10 that correspond to 1 Gbps and 300 Mbps data rates, respectively, we note that, the latency is much higher at 1 Gbps than at 300 Mbps. This is because the number of packets accumulated at the RLC buffer is indeed bigger, resulting in a more pronounced buffer bloat effect.

To overcome the buffer-bloat problem, we reduced the size of the RLC buffer to 5 MB (Fig 11). The throughput crashed, as the buffer cannot keep up with the packet arrival

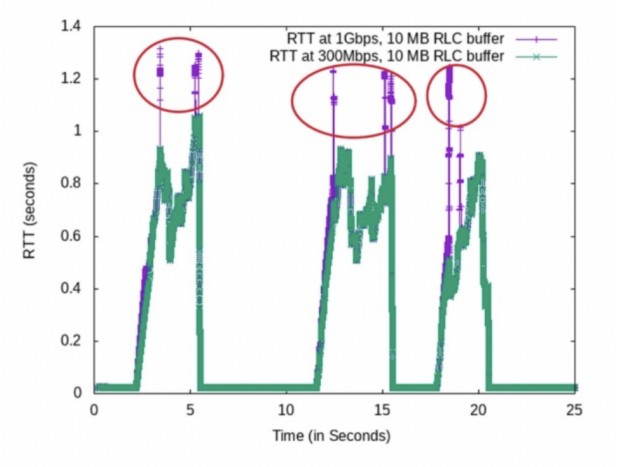


Fig. 10. Latency at 1Gbps and 300 Mbps Rates

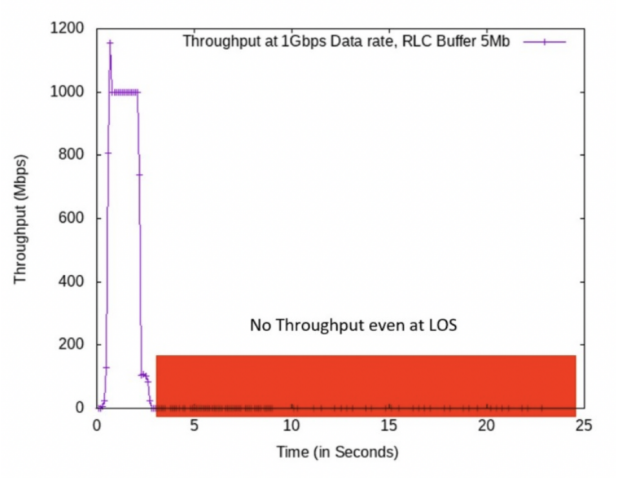


Fig. 11. Effect of Lower Buffer Size to Reduce Latency.

rate enabled by the large TCP congestion window and drops a large burst of packets. Consequently, TCP enters a fast-re-transmit phase, where 1 packet is re-transmitted every RTT. Thus, reducing buffer to address buffer bloat problem is not practical solution beyond a point.

### *B. Simulation of major TCP Congestion Control Algorithms in mmWave setup*

Set-up used for simulation is described in detail in System design -Environment

*1)* ***TCP New Reno:*** **Algorithm for TCP New Reno:** TCP New Reno has been the default algorithm for most communication systems. In the congestion avoidance phase, the congestion window cwnd is updated after the reception of every ACK. The update is based on the Additive Increase Multiplicative Decrease (AIMD) design: cwnd is increased by summing a term [$\alpha$/cwnd] for each received ACK and divided by a factor for each packet loss. For New Reno, these parameters are fixed to $\alpha = 1$ and $\beta = 2$.

The set-up is as defined in the System Design Section.

Fig 12 shows variation of throughput in LOS and NLOS regions. As expected, TCP New Reno aggressively attains maximum possible throughput in LOS but as soon as it encounters duplicate ACK it is reducing the window size by half. On encountering packet loss, it reduces window size almost to negligible, hence there is no throughput. Since New Reno is loss based, it detects congestion only upon receiving duplicate ACK or packet loss unlike TCP Vegas, hence due to buffer bloat it might experience some latency.

In Fig 13 and 14 RTT vs Time and Congestion Window graphs related how delays in RTT (implying congestion) are ultimately leads to lowering of congestion window.

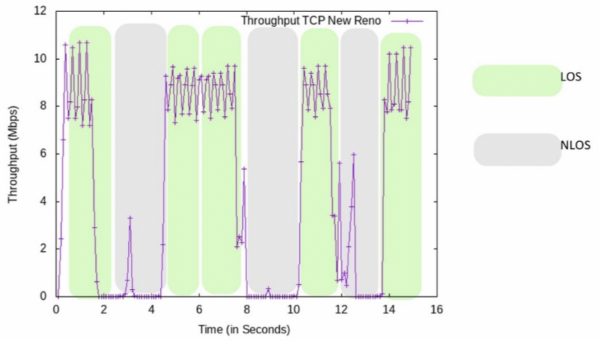


Fig. 12. Throughput Observation of TCP New Reno

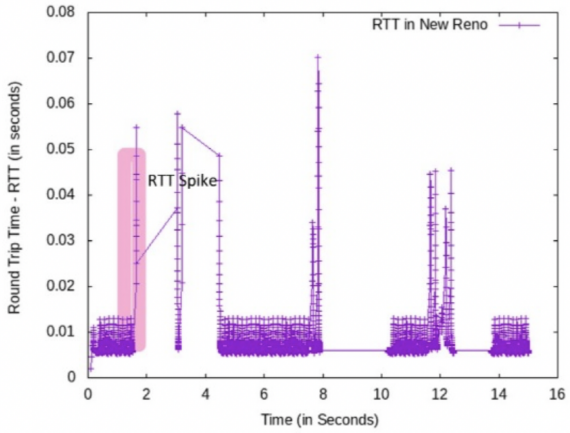


Fig. 13. RTT Observation of TCP New Reno

*2)* ***TCP Vegas:*** In comparison to TCP Reno, TCP Vegas uses delay(RTT) as a congestion indication in addition to the timeouts and duplicate acknowledgements. This makes it a delay based congestion control protocol which is more reliable in sensing the congestion in the underlay network.

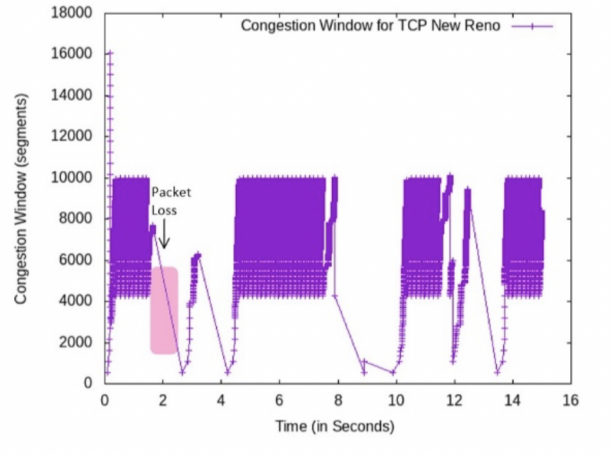


Fig. 14. Congestion Window for TCP New Reno

**RTT calculation:** When Vegas sends a packet, it stores the packet identification number and the system clock time stamp at that instant. After receiving the packet, it again reads the time stamp and compares it with the sent timestamp in order to calculate RTT. Therefore it uses a more accurate timer.

Because of this more accurate RTT calculation, Vegas can identify the congestion even before New Reno, which has to wait for three duplicate ACKs or timeouts.

This results in a more proactive approach in determining the congestion point rather quickly and provides more time to adjust the congestion window size. Therefore, we can avoid packets being dropped by reacting very quickly.

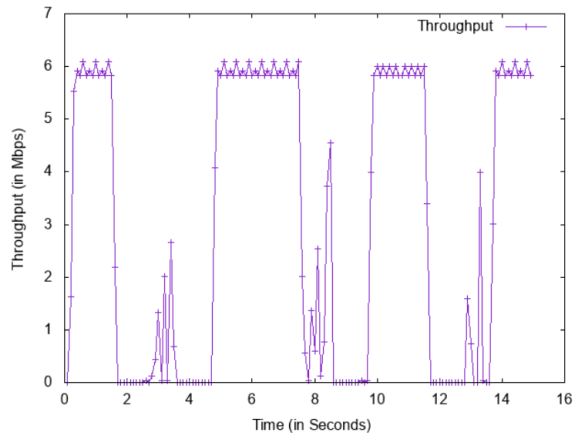


Fig. 15. Throughput Observation for TCP Vegas

It can be observed that the fluctuations in the throughput are far less compared to any other TCP protocol our switching engine has. This is because Vegas is able to detect congestion way before the other TCPs . Thus it makes sure not to increase and decrease the congestion window drastically over a short period of time. This provides a steady throughput. Thus, as per the application requirements we are having a steady throughput over a long period of time. However, the TCP Vegas's throughput is slightly worse than the other congestion based protocols.

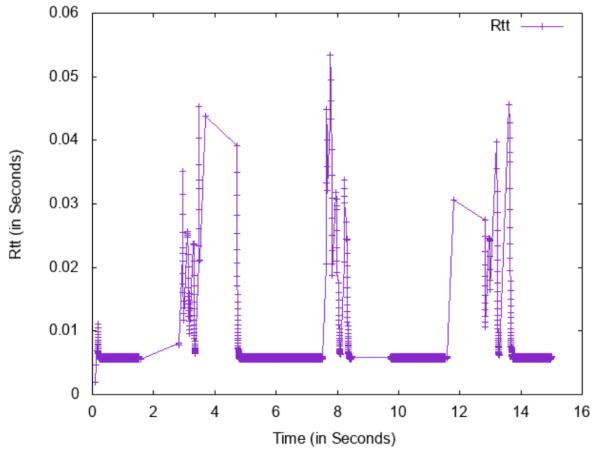


Fig. 16. RTT Observation for TCP Vegas

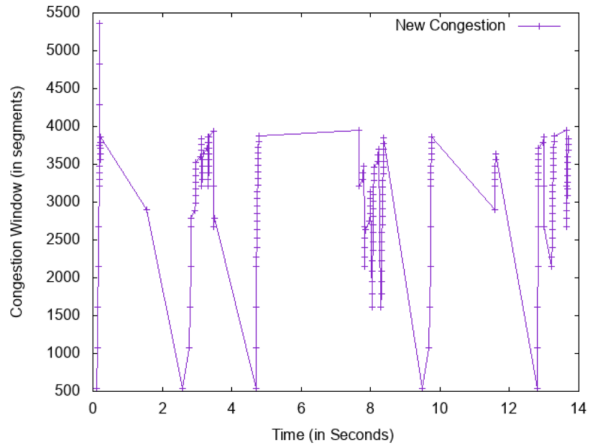


Fig. 17. Congestion Window for TCP Vegas

*3) **TCP Veno**:* When compared to TCP Reno, it has a refined multiplicative decrease algorithm that performs smart window adjustment based on the current state estimation of the connection rather than "blindly" halving the window after the fast re-transmit is triggered. This makes it able to send more packets compared to other TCP algorithms. Refer to Figure 18.

When a packet loss occurs, the veno uses the value of N, in order to determine the network conditions and adjusts its congestion window size, which makes it less aggressive compared to other loss based TCPs. This lets it attain a better throughput in case of NLOS communication links. Refer to

if $N_{queue} < \beta$, cwnd = cwnd + 1 each RTT

if $N_{queue} \geq \beta$, cwnd = cwnd + 0.5 each RTT

Fig. 18.

Figure 19.

if $N_{queue} < \beta$, the loss is probably *not* due to congestion; set cwnd = (4/5)×cwnd

if $N_{queue} \geq \beta$, the loss probably *is* due to congestion; set cwnd = (1/2)×cwnd as usual

Fig. 19.

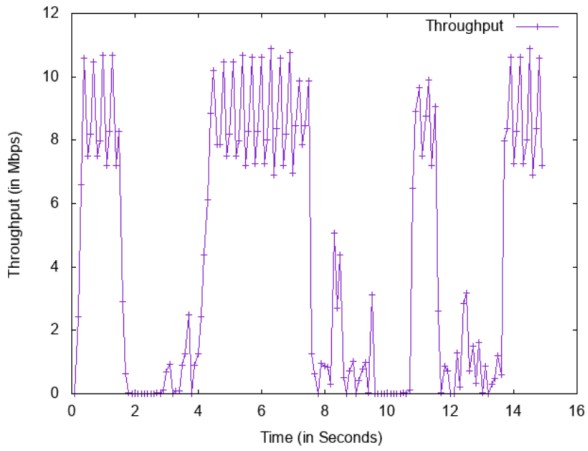


Fig. 20. Throughput Observation for TCP Veno

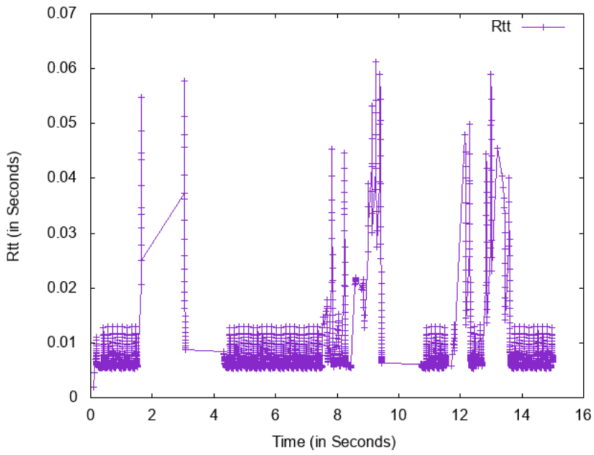


Fig. 21. RTT Observation for TCP Veno

It can be observed that Veno is trying to send packets

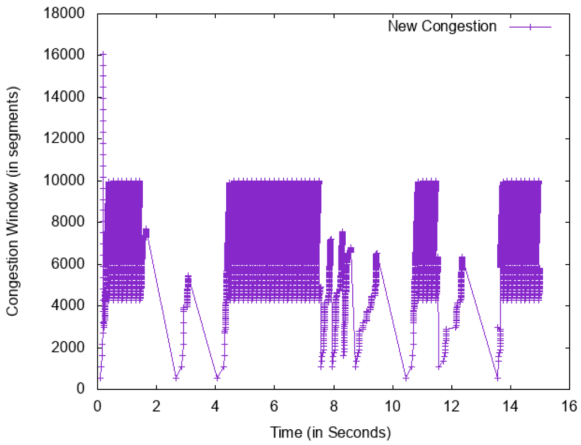


Fig. 22. Congestion Window for TCP Veno

even in the NLOS(Non-Line of Sight) Region. This is possible because it is adjusting its congestion window less aggressively when it detects a high-lossy medium which results in the reduction of the frequency of the packet drops and resetting to the slow start phase

*4) **TCP Cubic**:* TCP Cubic is yet another congestion control protocol which was introduced for high bandwidth and high latency networks. Cubic uses a cubic function instead of the linear window increase function. This allows the protocol to improve scalability and stability under high-speed and long-distance networks.

In our simulation, the application which requires high throughput and does not care about the time it takes to complete the data transfer chooses this TCP. In particular, for downloading large data from the remote server, TCP Cubic is selected because of its characteristics.

**Algorithm for TCP Cubic:** TCP Cubic behaves in the same manner as every other loss-based TCP. It updates the congestion window on receiving either positive acknowledgement or negative acknowledgement. Specifically, Cubic increases its window based on real-time and not based on RTT like Vegas. [26]

1. Window increase function:

$$cwnd(t) = C(T - K)^3 + Wmax \tag{1}$$

Where,

$$K = \sqrt[3]{\frac{Wmax\beta}{C}} \tag{2}$$

cwnd: current congestion window
T: Time elapsed since the last window reduction
C: Scaling constant
Wmax: max window size just before last reduction
: Multiplicative decrease factor

2. Concave Region:

When a ACK is received in congestion avoidance phase, and window is less than Wmax then TCP Cubic is in concave region. In the region, congestion window is incremented by:

$$window = (cwnd(t + RTT) – window)/window \quad (3)$$

3. Convex Region: When a ACK is received in congestion avoidance phase, and window is more than or equal to Wmax then TCP Cubic is in convex region. In the region, congestion window is incremented by:

$$window = (cwnd(t + RTT) – window)/window \quad (4)$$

4. Multiplicative Decrease: When there is packet loss due to duplicate ACKs, TCP Cubic updates Wmax, window and ssthresh as shown below;

Wmax = window;
ssthresh = window * $\beta$;
ssthresh = max(ssthresh, 2);
window = window * $\beta$;

5. Fast Convergence: To make TCP Cubic fair in terms of bandwidth sharing, convergence speed is improved. This makes the protocol to react quickly and give up some of the bandwidth if a new flow is introduced in the network. The quick reaction is made possible by lowering the congestion window and it is done as follows. To keep the last window before drop recorded, we use another variable Wpre_max. Refer to Figure 23.

```
If (Wmax < Wpre_max)
{
    Wpre_max = Wmax;
    Wmax = Wmax * (1 + β) / 2;
}
Else {
    Wpre_max = Wmax;
}
```

Fig. 23. Code

In mmWave network, TCP Cubic does not preform to its expectation. Since the protocol was designed for wired network in particular and also it aggressively increases the congestion window once packet is dropped. These characteristics affect the efficiency of the protocol in varying channel. Following figures show how the congestion window, RTT and Throughput behave in mmWave network. According to the section 4, we saw that the UE is moving from LoS to NLos and back to LoS for3 times. At around 1.5 seconds, the UE enters in the NloS for the first time. During this transition, UE is losing connectivity from the eNodeB. Hence, the packet is taking more time to travel and then the Ack for the same packet is received with a greater RTT than the previous packet. This goes on for few milliseconds until the connectivity is completely lost. Because of increased RTT, timeout occurs and in the corresponding congestion window graph we can see the window is dropped drastically, thus starting slow start phase. After the ssthresh, the congestion avoidance starts where then window is increasing gradually until the connectivity is lost. In the same period, the throughput is reduced and minimum amount of it is acquired until the UE is completely out of reach. This same behaviour you can observe two times. Once the UE exits NLoS region, the connectivity is back again and slow start phase starts and congestion is increased which triggers high throughput.

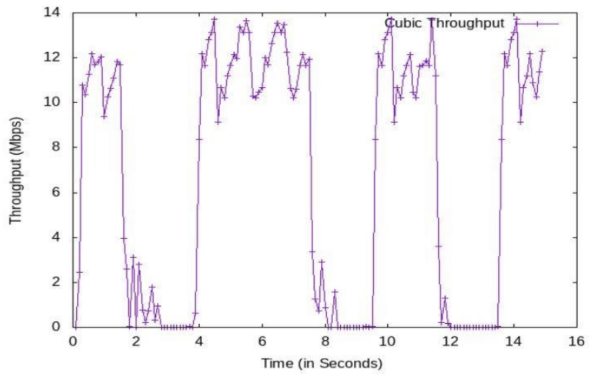


Fig. 24. Throughput Observation for TCP Cubic

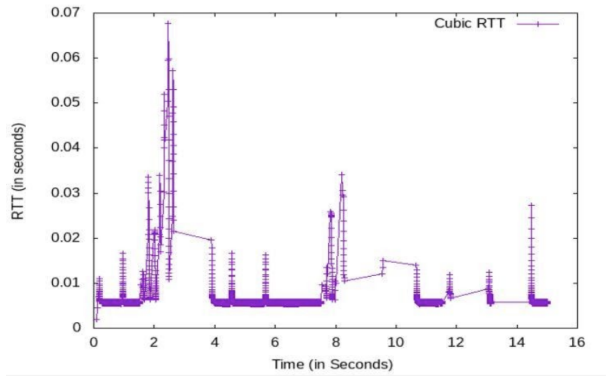


Fig. 25. RTT Observation for TCP Cubic

*5)* ***H-TCP****:* H-TCP is another loss-based algorithm. The aim of H-TCP is to effectively increase the aggressiveness of the protocol on high bandwidth-delay product paths and at the same time provide fairness on low bandwidth-delay product paths.

Application which require high bandwidth at the cost of latency chooses this algorithm. In particular, mobile internet uses this protocol, as it requires high bandwidth and needs to maintain fairness between multiple flows.

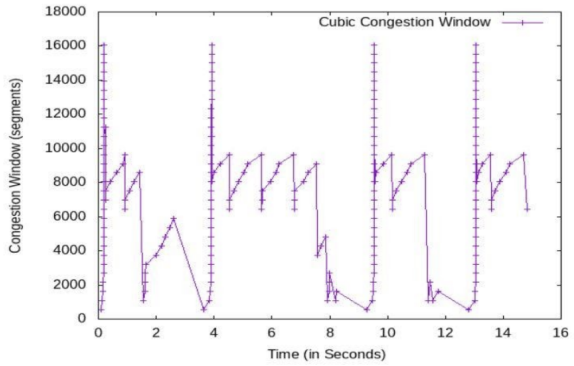


Fig. 26. Congestion Window for TCP Cubic

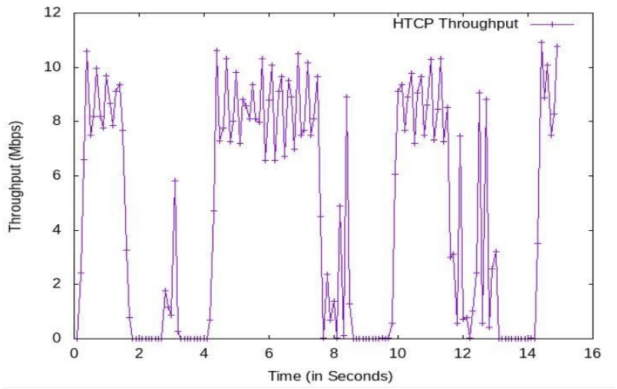


Fig. 27. Throughout Observation for H-TCP

**Algorithm for H-TCP:** The algorithm attempts to provided fairness between two or more flows by working mainly after packet loss. The algorithm uses back-off mechanism which is multiplicative. This mechanism aims to bring the bandwidth utilization of all flows to same level as quickly as possible. To achieve this instance, the algorithm drops the larger congestion window more in proportion and drops the smaller congestion window less in proportion, so that after a while both the flows start sharing bandwidth equally. [14]

$$w_i(k+1) = \beta w_i(k) + \alpha_i(k) T(k) \tag{5}$$

Where,

$\alpha_i$ (k) = 1 / RTT$_i$
$\omega_i$: congestion window of i flow
$\beta$: AIMD decrease parameter
$\alpha_i$ (k): AIMD increase parameter
T(k): interval between congestion events k and k+1

According to the section 4, we saw that the UE is moving from LoS to NLos and back to LoS for3 times. At around 1.5 seconds, the UE enters in the NloS for the first time. During this transition, UE is losing connectivity from the eNodeB. Hence, the packet is taking more time to travel and then the Ack for the same packet is received with a greater RTT than the previous packet. This goes on for few milliseconds until the connectivity is completely lost. Because of increased RTT, timeout occurs and in the corresponding congestion window graph we can see the window is dropped drastically, thus starting slow start phase. After the ssthresh, the congestion avoidance starts where then window is increasing gradually until the connectivity is lost. In the same period, the throughput is reduced and minimum amount of it is acquired until the UE is completely out of reach. This same behaviour you can observe two times. Once the UE exits NLoS region, the connectivity is back again and slow start phase starts and congestion is increased which triggers high throughput.

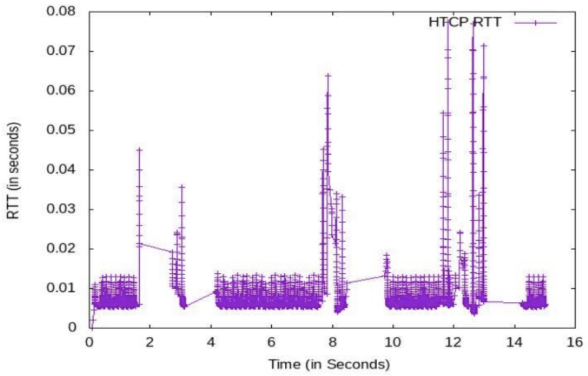


Fig. 28. RTT Observation for H-TCP

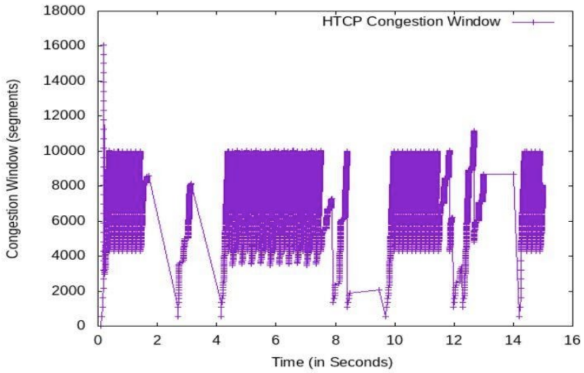


Fig. 29. Congestion Window for H-TCP

## VII. Conclusion

To understand and evaluate various TCP congestion control, we have simulated various TCP protocols on mmWave test bench on ns3 based simulation framework.

Our study of literature and simulations from mmWave module in ns3 revealed several potential problems in existing

congestion control algorithms of TCP.

As we have shown in Fig 8 because of slow start mechanism, an application might take several seconds to achieve full throughput offered by mmWave physical layer and thus create issues if they rely on short TCP connections. Since, packet drops will be a common scenario in NLOS region, latency could creep in due to queuing and buffering delays (Fig 9).

While RLC and MAC layer re-transmissions can protect upper layers from packet losses but in case of outage or re-transmission timeout (due to non LOS), even aggressive TCP protocols like Cubic can take long delays to fully recover to potential high data rates offered by mmWave Networks.

In section B of VI, we have demonstrated how various loss based and delay-based TCP congestion control reacts in mmWave network. Both the approaches have their own advantages and disadvantages. Aggressive loss-based congestion control algorithms like TCP New Reno and Cubic will give good throughput but in varying channel conditions and NLOS conditions due to queuing delays in buffers will have more latency. So, they would be suited for bandwidth guarantee class of applications which require minimum data rate for user satisfaction like video streaming on YouTube. However, for delay sensitive applications (Voice over IP) where latency is critical for user experience, aggressive congestion control is less suited as it experience more delays due to buffer bloat problem as shown in figure 10. In this scenario, delay-based congestion control mechanism like TCP Vegas or a congestion control mechanism which alters congestion window based upon channel characteristics is more suited (TCP Verus or TCP BBR). The only issue with these congestion controls is that they might need cross layer optimization which makes them less portable. Hence, we concluded that there is no one fit all congestion mechanism suited for mmWave networks. Hence, we explored the possibility of **Multi-TCP Application Aware Optimization** in mmWave Networks in consultation and under the guidance of professor.

We have seen MPTCP implementation in Figures 1 and 2 that show a lot of potential to address the issues mentioned above. MPTCP could, therefore, be integrated with DC and the 5G protocol stack. MP-TCP with the coupled BALIA CC algorithm fails to meet the first target of MP-TCP design, because in many cases its throughput is lower than that of SP-TCP. The uncoupled congestion control algorithm is unaffected by this issue because each path behaves independently. However, in this case the MP-TCP flow might be unfriendly to SP-TCP flows on shared bottlenecks. Due to paucity of time, we could not integrate MPTCP module in NS-3 and test it with our framework. We would like to extend this to our future work. In addition to it for more practical analysis, we plan to incorporate combination of diffuse scatter measurements and raytracing into our end-to-end mmWave simulation framework. We will incorporate MPTCP support.

## REFERENCES


[1] Jeremiah Abolade, Fakolujo A., and Abidemi Orimogunje. Handover in mobile wireless communication network - a review. *International Journal of Advanced Engineering, Management and Science*, 3:934–940, 01 2017.

[2] Jong Ahn, Peter Danzig, Z. Liu, and L. Yan. Experience with tcp vegas: emulation and experiment. 01 1995.

[3] Eneko Atxutegi, Fidel Liberal, Habtegebreil Haile, Karl-johan Grinnemo, Anna Brunstrom, and Åke Arvidsson. On the use of tcp bbr in cellular networks. *IEEE Communications Magazine*, 56, 03 2018.

[4] Tommy Azzino, Matteo Drago, Michele Polese, Andrea Zanella, and Michele Zorzi. X-tcp: a cross layer approach for tcp uplink flows in mmwave networks. pages 1–6, 06 2017.

[5] M. Bennis, M. Debbah, and H. V. Poor. Ultrareliable and low-latency wireless communication: Tail, risk, and scale. *Proceedings of the IEEE*, 106(10):1834–1853, Oct 2018.

[6] Carlo Caini and Rosario Firrincieli. Tcp hybla: a tcp enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking*, 22:547 – 566, 09 2004.

[7] Geetanjali Chellani and Anshuman Kalla. A review: Study of handover performance in mobile ip. *International journal of Computer Networks Communications*, 5, 12 2013.

[8] Wu-chang Feng, Dilip Kandlur, and Debanjan Saha. The blue active queue management algorithms. *Networking, IEEE/ACM Transactions on*, 10:513– 528, 09 2002.

[9] Cheng Fu and Soung Chang Liew. Tcp veno: Tcp enhancement for transmission over wireless access networks. *Selected Areas in Communications, IEEE Journal on*, 21:216 – 228, 03 2003.

[10] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *Operating Systems Review*, 42:64–74, 07 2008.

[11] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. An in-depth study of lte: Effect of network protocol and application behavior on performance. volume 43, pages 363–374, 08 2013.

[12] XiuJia Jin. A survey on network architectures for mobility. 03 2020.

[13] Esha Kumar, Shivanka Chugh, and S.P. Ghrera. Performance analysis of red with different tcp congestion control mechanism. pages 414–417, 04 2011.

[14] Douglas Leith and R. Shorten. H-tcp: Tcp for high-speed and long-distance networks. 03 2004.

[15] Xiaolan Liu, Feng yuan Ren, Ran Shu, Tong Zhang, and Tao Dai. Mitigating bufferbloat with receiver-based tcp flow control mechanism in cellular networks. 2015.

[16] Saverio Mascolo, Claudio Casetti, Mario Gerla, M.Y. Sanadidi, and Ren Wang. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. volume 8, pages 287–297, 01 2001.

[17] P. J. Mateo, C. Fiandrino, and J. Widmer. Analysis of tcp performance in 5g mm-wave mobile networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–7, May 2019.

[18] Sherif Moussa, Mohammad Wahba, and Salah Abdel-Mageid. Performance evaluation of snoop protocol for wireless networks. *Journal of Advances in Computer Networks*, 3:124–127, 01 2015.

[19] I. Petrov and T. Janevski. Advanced 5g-tcp: Transport protocol for 5g mobile networks. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 103–107, Jan 2017.

[20] Ivan Petrov and Toni Janevski. Design of novel 5g transport protocol. pages 29–33, 10 2016.

[21] Ivan Petrov and toni janevski. Advanced 5gtcp: Transport protocol for 5g mobile networks. 01 2017.

[22] M. Polese, R. Jana, and M. Zorzi. Tcp and mp-tcp in 5g mmwave networks. *IEEE Internet Computing*, 21(5):12–19, 2017.

[23] Michele Polese, Federico Chiariotti, Elia Bonetto, Filippo Rigotto, Andrea Zanella, and Michele Zorzi. A survey on recent advances in transport layer protocols, 10 2018.

[24] Michele Polese, Rittwik Jana, and Michele Zorzi. Tcp in 5g mmwave networks: Link level retransmissions and mp-tcp. 03 2017.

[25] Costin Raiciu, Christoph Paasch, Sébastien Barré, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? designing and implementing a deployable multipath tcp. pages 29–29, 04 2012.

[26] Injong Rhee, Lisong Xu, Sangtae Ha, Alexander Zimmermann, Lars Eggert, and Richard Scheffenegger. Cubic for fast long-distance networks. *RFC*, 8312:1–18, 2018.

[27] Jaydip Sen. *Mobility and Handoff Management in Wireless Networks*, pages 457– 484. 03 2010.

[28] Murtaza Siddiqi, Jaehyung Yu, and Joung. 5g ultra-reliable low-latency communication implementation challenges and operational issues with iot devices. *Electronics*, 8:981, 09 2019.

[29] Prasun Sinha, Narayanan Venkitaraman, Raghupathy Sivakumar, and Vaduvur Bharghavan. Wtcp: a reliable transport protocol for wireless wide-area networks. In *MobiCom '99*, 1999.

[30] N. D. Tripathi, J. H. Reed, and H. F. VanLandinoham. Handoff in cellular systems. *IEEE Personal Communications*, 5(6):26–37, Dec 1998.

[31] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *NSDI*, 2013.

[32] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive congestion control for unpredictable cellular networks. *ACM SIGCOMM Computer Communication Review*, 45:509–522, 08 2015.

[33] M. Zhang, M. Mezzavilla, R. Ford, S. Rangan, S. Panwar, E. Mellios, D. Kong, A. Nix, and M. Zorzi. Transport layer performance in 5g mmwave cellular. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 730–735, April 2016.

[34] Menglei Zhang, Marco Mezzavilla, Jing Zhu, Sundeep Rangan, and Shivendra Panwar. The bufferbloat problem over intermittent multi-gbps mmwave links. 11 2016.

[35] F. Zhou, D. Choffnes, and K. Chowdhury. Janus: A multi-tcp framework for application-aware optimization in mobile networks. *IEEE Transactions on Mobile Computing*, 18(9):2103–2116, 2019.