

Boosting TCP & QUIC Performance in mmWave, Terahertz, and Lightwave Wireless Networks: a Survey

E. Khorov^{*†}, A. Krasilov^{*†}, M. Susloparov^{*†}, L. Kong[‡]

^{*}HSE University, Moscow, Russia

[†]Institute for Information Transmission Problems of the Russian Academy of Sciences, Moscow, Russia

[‡]Shanghai Jiao Tong University, Shanghai, China

E-mail: {ekhorov, akrasilov, msusloparov}@hse.ru, linghe.kong@sjtu.edu.cn

Abstract—Emerging wireless systems target to provide multi-Gbps data rates for each user, which can be achieved by utilizing ultra-wide channels available at mmWave, terahertz, and lightwave frequencies. In contrast to the traditional spectrum below 6 GHz, these high-frequency bands raise many issues, complicating their usage. For example, because of high signal attenuation and blockage by obstacles, the data rates in a high-frequency band may quickly vary by several orders of magnitude. This peculiarity is often considered a challenge for modern transport layer protocols, such as Transmission Control Protocol (TCP) or Quick UDP Internet Connections (QUIC). Their key component is the Congestion Control Algorithm (CCA), which tries to determine a data sending rate that maximizes throughput and avoids network congestion. Many recent studies show that the performance of the existing CCAs significantly degrades if mobile devices communicate with high-frequency bands and propose some solutions to address this problem. The goal of this survey is twofold. First, we classify the reasons for poor TCP & QUIC performance in high-frequency bands. Second, we comprehensively review the solutions already designed to solve these problems. In contrast to existing studies and reviews that mainly focus on the comparison of various CCAs, we consider solutions working at different layers of the protocol stack, i.e., from the transport layer down to the physical layer, as well as cross-layer solutions. Based on the analysis, we conclude the survey with recommendations on which solutions provide the highest gains in high-frequency bands.

Index Terms—TCP, QUIC, Wi-Fi, 5G, 6G, wireless networks, mmWave, Terahertz, Lightwave communications

I. INTRODUCTION

Wireless communication technologies continuously evolve, providing higher and higher data rates to end users. Modern Wi-Fi and cellular systems that operate in traditional low-frequency bands (i.e., below 6 GHz) provide goodput up to several hundreds of Mbps per user [1]. The next frontier is to achieve multi-Gbps data rates, which are required to support emerging applications, such as augmented/virtual reality, cloud gaming, eHealth, autonomous transportation systems, etc. Since the low-frequency bands are overpopulated, a promising way to achieve this goal is to use wide channels available in mmWave (30 GHz to 300 GHz), Terahertz (300

GHz to 3 THz), Infrared (300 THz to 420 THz) and Visible Light (420 THz to 750 THz) bands. In particular, recent Wi-Fi amendments [2] and the fifth generation (5G) cellular systems [3] already use mmWave bands for data transmission. IEEE 802.11 Working Group is finalizing its work on the IEEE 802.11bb standard aka Li-Fi, which extends Wi-Fi to Lightwave (LW) bands [4], [5]. High frequencies are candidates for various future wireless communication systems [6]–[8].

A common drawback of these high-frequency bands is extremely high propagation loss and blockage by obstacles, which: (i) shrinks the coverage and (ii) makes data rates quickly fluctuate. In particular, an obstacle appearing between a mmWave transmitter and a receiver can instantly drop the data rate from several Gbps down to dozens of Mbps [9], [10]. For higher frequencies, e.g., LW, the link can be completely blocked. To overcome these issues, Wi-Fi and 5G systems can jointly use low-frequency bands that provide continuous coverage and high-frequency bands to boost data rate when the channel conditions are favorable [11]–[13]. For example, New Radio (NR) — a new radio access technology (RAT) developed by the 3rd Generation Partnership Project (3GPP) — employs both traditional low-frequency bands below 6 GHz and high-frequency bands from 24 to 71 GHz. Aggregating up to 16 channels¹, each of 400 MHz width (i.e., the total bandwidth of 6.4 GHz) available in the mmWave band, NR can reach, in theory, 140 Gbps [14], which is far beyond the 5G requirements [15]. Similarly, Wi-Fi can jointly use 2.4, 5, 6 GHz, mmWave, and LW bands [5]. No doubt that future devices supporting THz bands will follow this paradigm.

Many existing mobile applications, e.g., web browsers, messengers, video players, and social networks clients use Transmission Control Protocol (TCP) to provide reliable and in-sequence communications between a mobile device and a server. A key component of TCP that significantly affects its performance is the Congestion Control Algorithm (CCA). This algorithm is implemented at the TCP sender and dynamically selects the data sending rate that maximizes throughput and

Support from the Basic Research Program of the National Research University Higher School of Economics is gratefully acknowledged.

¹The base station supports up to 16 channels. Due to lower processing capabilities and battery power supply, mobile devices use only a few channels.

avoids congestion. That is why most of the studies related to TCP are focused on designing new congestion control algorithms or their enhancements (see Section VI-A for details).

Recently Google has started a campaign to replace TCP with a new protocol called Quick UDP Internet Connections (QUIC). QUIC was developed as an experimental protocol to improve the Chrome browser's performance and then submitted to Internet Engineering Task Force (IETF) [16]. Now QUIC is implemented in almost all modern web browsers, and approximately 9% of websites support communication via QUIC [17]. Despite architectural changes, QUIC implements congestion control ideas similar to TCP. Consequently, it inherits many peculiarities of TCP and only slightly improves goodput, as reported by numerous studies, e.g., [18]–[21].

Initially, TCP developers considered only wired networks where the endpoint devices typically have access links (i.e., the links between the client devices and the network) with almost constant capacity. However, today the largest part of the Internet traffic belongs to wireless clients [1]. In contrast to wired links, the capacity of wireless ones significantly changes with time because of the fluctuation of both the channel quality and the number of devices sharing the same channel. As the data rates in the wireless networks are typically lower than in the wired ones, the wireless link becomes a bottleneck on the path between the device and the server and, thus, limits the performance of the corresponding TCP or QUIC connection. That is why during the last three decades, IETF — the main developer of TCP — and other researchers have proposed many algorithms and protocol modifications to improve TCP & QUIC performance in wireless networks [16], [22], [23].

Many recent studies [24]–[29] show that the properties of mmWave bands (such as highly fluctuating data rates, frequent link quality drops and path changes, etc.) significantly degrade the performance of TCP & QUIC. While the authors of these papers develop new solutions aimed to improve TCP & QUIC performance in particular scenarios, our paper investigates general problems arising in high-frequency wireless systems. Although the problems inherent to mmWave, THz, and LW bands have much in common, the observed effects, as well as the solutions, may differ. Specifically, the obstacles which only degrade the mmWave channel quality may completely block the LW signal. On one side, the usage of higher frequencies leads to wider channels and higher data rates. On the other side, as higher frequencies reduce the coverage, we need much more THz or LW base stations to cover a given area. Consequently, they shall be cheap. Both high data rates and low costs limit the complexity of the implemented approaches to improve TCP & QUIC performance. This survey takes into account these peculiarities and studies: (i) whether the identified problems can be solved with the existing approaches or whether we need to develop new ones, (ii) which approaches can work with TCP & QUIC and which limitations are caused by specific bands, (iii) which approaches provide the best performance.

Recent surveys [30]–[34] confirm the research community's high interest in enhancing the performance of transport layer

TABLE I: List of acronyms.

3GPP	3rd Generation Partnership Project
5G	Fifth Generation
6G	Sixth Generation
ACK	Acknowledgment
AIMD	Additive Increase/Multiplicative Decrease
ANDSF	Access Network Discovery and Selection Function
AQM	Active Queue Management
BDP	Bandwidth Delay Product
CA	Congestion Avoidance
CAPEX/OPEX	Capital expenditures/Operational expenses
CDN	Content Delivery Network
CoMP	Coordinated Multi-Point
CWND	Congestion Window
DNS	Domain Name System
ECN	Explicit Congestion Notification
FIFO	First-In-First-Out
FST	Fast Session Transfer
HTTP	HyperText Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
LW	LightWave
LOS	Line of Sight
MAC	Medium Access Control
MC	Multi-Connectivity
MEC	Multi-access Edge Computing
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MP	Multipath
MSS	Maximum Segment Size
NFV	Network Functions Virtualization
NLOS	Non-Line of Sight
NR	New Radio
OS	Operating System
OSI	Open Systems Interconnection
PDCCP	Packet Data Convergence Protocol
PEP	Performance Enhancing Proxy
PGW	Packet Gateway
PHY	Physical Layer
PTO	Probe TimeOut
RAT	Radio Access Technology
RIS	Reconfigurable Intelligent Surface
RL	Reinforcement Learning
RTO	Retransmission TimeOut
RTT	Round Trip Time
RWND	Receive Window
SNR	Signal to Noise Ratio
SSThr	Slow Start Threshold
TCL	Traffic Control Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
QoS	Quality of Service
QUIC	Quick UDP Internet Connections
UDP	User Datagram Protocol
UE	User Equipment

protocols in wireless systems operating in high-frequency bands. In particular, [30] reviews recently developed solutions aiming to improve the performance of transport layer protocols in modern wired/wireless systems. Surveys [31], [32] analyze TCP and multi-path TCP performance issues arising in mmWave systems and consider various solutions operating at the transport layer that can address them. Authors of [33], [34] review a wide range of congestion control algorithms and study which of them provide the best performance in 5G wireless systems. While the mentioned surveys focus on the solutions working only at the transport layer (e.g., congestion control algorithms, multi-path TCP & QUIC extensions), we

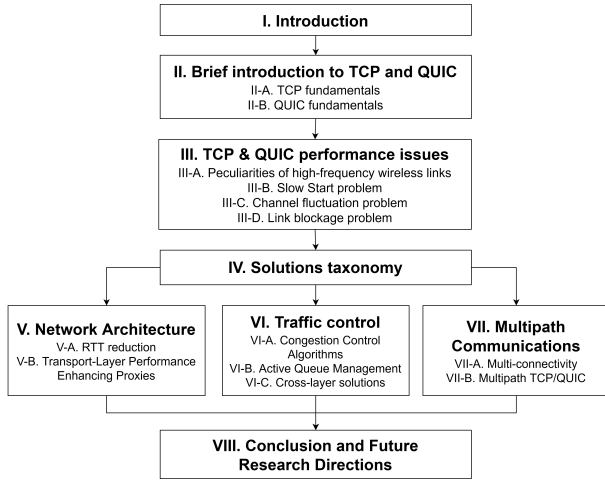


Fig. 1: The structure of the survey.

consider solutions working at different layers of the protocol stack (from the transport layer down to the physical one) and also consider cross-layer solutions. Moreover, in contrast to the existing works that consider only the mmWave band, we analyze how various solutions can be applied to higher frequency bands (i.e., THz and LW bands).

The contributions of the survey are as follows:

- 1) We investigate and classify the problems leading to poor TCP & QUIC performance in high-frequency wireless systems and show that these problems are common for all mmWave, LW, and THz bands, though the scale of the problems may differ.
- 2) We analyze the existing solutions to these problems proposed by IETF, 3GPP, IEEE, and the research community. Note that, typically, these solutions are applicable to all types of high-frequency bands. Otherwise, we emphasize band-related issues.
- 3) We enrich the analysis of various solutions with numerical results obtained under the same conditions to compare them quantitatively rather than only qualitatively.
- 4) We discuss the implementation complexity of various solutions and provide recommendations for network operators and service providers on which of them shall be used to achieve high TCP & QUIC performance.

The rest of the survey is organized as follows (see Fig. 1). In Section II, we provide a brief introduction to TCP and QUIC. In Section III, we consider typical scenarios for high-frequency wireless systems and study the problems that lead to TCP & QUIC performance degradation. Section IV contains a taxonomy of various solutions aiming to improve TCP & QUIC performance, while Sections V–VII provide their detailed description. Finally, in Section VIII, we summarize our findings and discuss the directions for future research. Table I lists used acronyms.

II. BRIEF INTRODUCTION TO TCP AND QUIC

In this section, we introduce TCP and QUIC — the two most widely used protocols that provide reliable commu-

nication between a mobile device and a server. Note that there exist other reliable transport protocols such as Stream Control Transmission Protocol (SCTP). However, SCTP has not obtained wide adoption on the mobile Internet because of: (i) the transport layer ossification problem and a low number of operating systems and applications supporting it [30], [35]. We start with the description of TCP in Section II-A. Then we discuss QUIC, paying more attention to the key QUIC novelties aimed to improve its performance compared to TCP.

A. TCP fundamentals

Since the early 1980s, when the first TCP specification [36] was published, many enhancements have been proposed to improve TCP performance in various scenarios. An interested reader can find a brief review of the TCP enhancements specified over the last 40 years by IETF in [22]. Some of these enhancements were widely deployed, i.e., they were implemented in various Operating Systems (OSs), while the others are still experimental. Note that various OSs may implement these enhancements differently.

In this section, we briefly review the key TCP features that are widely deployed (i.e., used by default in various OSs) and describe what can be called “standard” TCP behavior. Non-standard (experimental) enhancements developed by IETF and/or the research community are considered in Section V–VII, focusing on those enhancements that improve TCP performance in high-frequency wireless networks.

TCP is a transport layer protocol that works at endpoints, e.g., a server located on the Internet uses TCP to send data to a User Equipment (UE) as shown in Fig. 3(a). Because of transport layer abstraction, the TCP sender considers the whole path between the server and the UE as the black box ignoring the peculiarities of particular links, e.g., a link between the base station and the UE. The goal of the TCP sender is to send such an amount of data that the network path is fully used without congestion (i.e., loss of data at the intermediate nodes because of overload). The whole network path can be characterized by two parameters: (i) Round Trip Time (RTT), which is the time between sending a packet and receiving an acknowledgment (ACK) for it, and (ii) bottleneck capacity C , i.e., the maximum data rate that can be served by the network without congestion. Once the sender generates traffic with a rate above C , the link with the lowest capacity along the path becomes overloaded, which leads to packet losses. Currently, the wireless link is typically a bottleneck², which requires the transport protocol to consider its peculiarities.

To control the sending rate, the TCP sender uses a sliding window approach and keeps an internal variable called Congestion Window (CWND) that limits the amount of data the TCP sender can send without obtaining ACKs for this data. In addition, the TCP receiver can indicate the size of its receive buffer (called Receive Window, RWND) in ACKs sent back to the TCP sender. According to [36], the TCP sender can send

²The capacity of backhaul links, i.e., the links connecting base stations and the core network, typically exceeds that of wireless links.

no more than $\min(\text{CWND}, \text{RWND})$ bytes without obtaining an ACK. Note that if no specific optimizations are used (see Section V), RWND is typically much higher than CWND. Thus, the sending rate is mainly limited by CWND.

Since ACKs arrive RTT after the transmission of data packets, the average sending rate of TCP connection can be estimated as W/RTT , where W is the size of CWND in bytes. If the rate is less than C , the network path is underutilized. Otherwise, if the sending rate is higher than C , the queue at the bottleneck link will continuously grow, eventually leading to packet losses (i.e., congestion). Therefore, the TCP sender should keep CWND as close as possible to $W_{\text{opt}} = C \cdot \text{RTT}$ called Bandwidth Delay Product (BDP).

While RTT can be measured at the TCP sender as the time between sending a packet and receiving an ACK for this packet, the capacity C of the bottleneck link is unknown at the TCP sender. The core function of the TCP sender called the congestion control algorithm is to properly select CWND in order to keep it as close as possible to BDP. In turn, BDP can vary with time because the wireless channel quality and/or the number of competing flows changes.

According to [37], any congestion control algorithm can be split into four algorithms: Slow Start, Congestion Avoidance (CA), Fast Retransmit, and Fast Recovery, see Fig. 2. Apart from CWND, the TCP sender keeps an internal variable called Slow Start Threshold (SSThr) to select which algorithm to use: if CWND is below SSThr, the TCP sender uses the Slow Start algorithm; otherwise, it uses the CA algorithm. When some packets are lost, the TCP sender uses Fast Retransmit and Fast Recovery algorithms to retransmit lost packets.

After the TCP connection is established, SSThr is set to a high value³ and the Slow Start algorithm is used to increase CWND to reach BDP. A detailed description of the Slow Start algorithm is provided in Section III-B. When CWND exceeds BDP, the buffer size at the bottleneck link increases, eventually leading to buffer overflow and packet losses. When the TCP sender detects that some packet is lost, it: (i) finishes the Slow Start algorithm, (ii) updates SSThr, (iii) reduces CWND to limit data sending rate, (iv) retransmits the lost packet (i.e., performs the Fast Retransmit algorithm), and (iv) enters the so-called Fast Recovery algorithm [37]. During the Fast Recovery algorithm, the TCP sender shall retransmit and, eventually, deliver all the lost packets. Additionally, the TCP sender can transmit new data packets to use the available link capacity. When all the packets considered lost are recovered, the TCP sender enters the CA algorithm.

Most of the widely used CA algorithms are based on Additive Increase Multiplicative Decrease (AIMD) principle that works as follows. The TCP sender gradually increases CWND until a congestion event occurs. For example, the widely used CUBIC algorithm [38] increases CWND with time as a cubic function until the congestion event occurs (see Fig. 2). After a congestion event, the TCP sender reduces

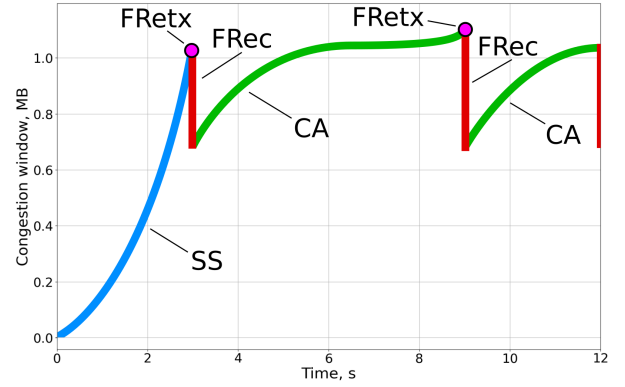


Fig. 2: CWND as the function of time for CUBIC. CWND is managed by the algorithms responsible for Slow Start (SS), Congestion Avoidance (CA), Fast Retransmit (FRetx), and Fast Recovery (FRec).

CWND by multiplying it by some parameter $\beta < 1$ and uses Fast Retransmit and Fast Recovery algorithms to recover lost packets. After the successful delivery of all lost packets, it enters the CA algorithm again. So, in the case of stable link characteristics and a high amount of transmitted data, the TCP sender runs the CA algorithm most of the time.

Because of the mentioned fact, during the last decades, the Slow Start algorithm remained almost unchanged, while much research was focused on finding the best CA algorithm. However, a wide variety of bottleneck links (e.g., wired, wireless, satellite) and a wide variety of network deployment scenarios do not allow finding a single algorithm that fits all cases. Moreover, in recent years, the appearance of ultra-high bandwidth mmWave links having highly fluctuating data rates has led to a novel cycle of the CA algorithms development, e.g., see [39]–[42]. In this survey, we analyze in detail the issues that cause poor TCP performance for the existing congestion control algorithms and analyze possible alternatives (see Section VI-A).

B. QUIC Fundamentals

TCP is implemented in the OS kernel. On the one hand, it can efficiently process thousands of parallel TCP connections, which is important for servers. On the other hand, even small protocol changes require OS kernel upgrades, which may take a long time and require much effort to make these changes widespread. This problem is known as the transport layer ossification [35]. The long TCP evolution process was one of the reasons why Google proposed a novel protocol called QUIC and then submitted it to IETF as an alternative to TCP [16], [43]. The key idea behind QUIC is to move the main TCP functions (connection management, congestion control, loss recovery) from the transport layer to the application layer. Being implemented in the user space (e.g., in browsers), QUIC can be easily modified, which significantly speeds up the protocol evolution and design of innovative solutions.

While, formally, QUIC operates at the application layer and uses the User Datagram Protocol (UDP) as the underlying

³If the server uses the data path for the first time, it sets SSThr to the infinite value. Otherwise, a previously saved value may be used [37].

transport layer protocol, QUIC itself is considered a transport protocol because it uses many features similar to TCP. Below, we focus on new features introduced in QUIC and discuss how they improve its performance with respect to TCP. The Internet-scale evaluation of QUIC done by Google can be found in [44].

1) *Connection management*: QUIC introduces a novel connection management procedure. Each QUIC connection is identified by a unique eight-byte number called Connection ID. When two end-points establish a new QUIC connection, they exchange service packets in which they negotiate a new Connection ID, security keys (obtained with the Transport Layer Security (TLS) protocol, which is integrated with QUIC), and other transmission parameters. This procedure lasts one RTT, and after it finishes, the end-points can transmit data over the secured connection. In contrast, TCP requires at least two RTTs to set up a new secure connection: one RTT to set up a TCP connection and one RTT to negotiate security keys because TLS works on top of TCP. If the end-points have already agreed on security parameters during previous QUIC connections, for subsequent QUIC connections, it is possible to send the user data in the very first packet. This feature is known as “0-RTT connection setup”, and it is very fruitful for applications that rarely send low amounts of data (e.g., machine-type communication, web search/assistance, etc.)

Another advantage of QUIC connection management is the resilience to frequent handovers, e.g., when users switch between Wi-Fi and cellular networks. If the networks belong to different operators, the user changes its Internet Protocol (IP) address. As QUIC operates at the application layer and a QUIC connection is identified by a Connection ID, a change of ports and addresses for the underlying IP and UDP protocols does not influence the QUIC connection. In contrast, when TCP is used and the client changes its IP address, all ongoing TCP connections shall be re-established, which takes much time. Moreover, the applications shall retransmit data from scratch.

2) *Header encryption*: As QUIC operates at the application layer, it uses TLS to encrypt both the header and payload. Only a few QUIC header fields (e.g., Flags, Connection ID) are transmitted unencrypted. QUIC developers argue that this feature is very important and protects QUIC connections from the negative influence of middleboxes, i.e., intermediate network nodes that modify packet headers to achieve local performance improvement. As shown in several studies (e.g., [44], [45]), if several middleboxes operate independently, they can apply inconsistent solutions resulting in poor overall performance.

In contrast to QUIC, all fields of the TCP header are open, which enables many popular performance-enhancing solutions that break TCP connections or modify the TCP header. In Section V-B, we analyze the benefits and drawbacks of such solutions and discuss which of them can be applied to QUIC.

3) *Multi-streaming*: QUIC supports concurrent transmission of several data streams between end-points. This feature is especially useful for web traffic that typically consists of multiple objects (e.g., text, scripts, pictures). QUIC can create

a separate stream for each object. Thus, each object can be independently transmitted, processed, and displayed to a user. This feature solves the Head-of-Line (HOL) blocking problem inherent to TCP [46]. Note that by design, TCP supports only a single byte stream between end-points. For this reason, if even a single packet belonging to one object is lost, the TCP receiver cannot process packets belonging to subsequent objects until the lost packet is recovered.

4) *Congestion control*: The general QUIC congestion control architecture specified [47] is similar to that of TCP [37]. In particular, a QUIC sender state machine consists of Slow Start, Congestion Avoidance, and Recovery states. The latter state combines Fast Retransmit and Fast Recovery used in TCP. Thus, a plethora of congestion control algorithms designed for TCP and described in detail in Section VI-A can be applied to QUIC. In particular, the default QUIC congestion control algorithms (i.e., NewReno, CUBIC) are inherited from TCP. Nevertheless, QUIC developers introduced several features aiming to enhance loss recovery with respect to TCP.

First, the QUIC sender assigns a unique sequence number to each sent packet. When the QUIC receiver creates an ACK for this packet, it includes this sequence number in the ACK header. Thus, the QUIC sender can easily infer which particular transmission is acknowledged. This feature allows resolving the so-called “retransmission ambiguity” problem inherent to TCP: in case of several transmissions of the same packet, the TCP sender cannot determine which particular transmission attempt is acknowledged.

Second, the QUIC header has a special field that contains a delay between the reception of the data packet at the QUIC receiver and the transmission of the corresponding ACK. This information and the unique packet sequence numbers allow the QUIC sender to obtain more accurate RTT estimation.

Third, QUIC changes the loss detection procedure. The TCP sender detects a packet loss when it receives three ACKs in a row with the same sequence number (called duplicate ACKs) or when the retransmission timeout (RTO) timer expires⁴. In the latter case, the TCP sender shrinks CWND to one packet and retransmits the packet for which RTO was initiated. Instead of RTO, QUIC introduces the probe timeout (PTO) mechanism. PTO is set based on RTT estimation. When PTO expires, the QUIC sender does not shrink the congestion window and sends the so-called probe packet (e.g., the previously unsent data packet or retransmission) that should be acknowledged. Only when persistent congestion is detected, the QUIC sender reduces the congestion window. This feature improves QUIC performance when tail data packets or ACKs are lost.

Many studies compare the performance of TCP and QUIC congestion control. The results show that the overall performance significantly depends on the considered scenario, used parameters, and even peculiarities of implementation [18]–

⁴RTO is set based on RTT estimation at the TCP sender. The minimal value of RTO according to RFC 6298 is 1s [48] while Linux implementations use 200ms [49].

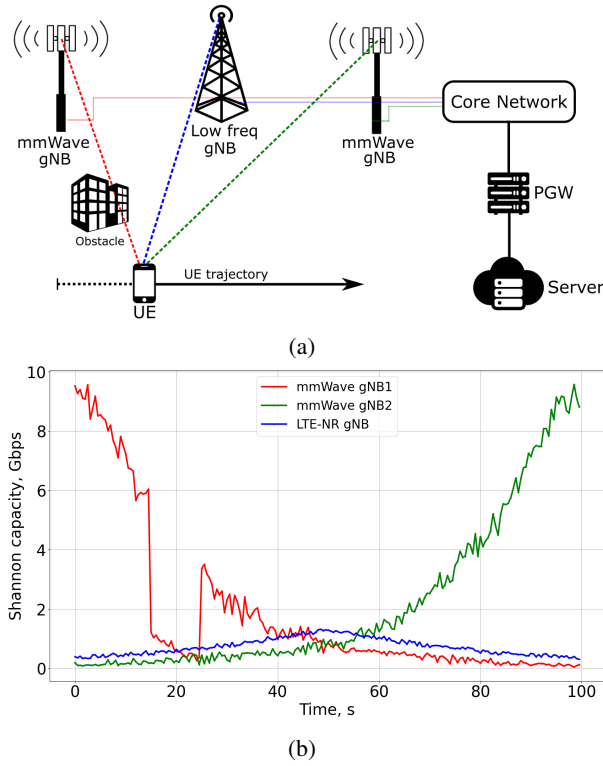


Fig. 3: Characteristics of high-frequency wireless links: (a) network with two mmWave gNBs operating in 1 GHz@28 GHz channel, one LTE/NR gNB operating in 100 MHz@3.6 GHz channel, (b) Shannon capacity of the corresponding wireless links.

[21]. So, there is now an unequivocal advantage of QUIC congestion control over TCP.

III. TCP & QUIC PERFORMANCE ISSUES

In contrast to low-frequency links, high-frequency ones may provide much higher but very unstable capacity, degrading TCP performance, as described below. Notably, being inherited from TCP, QUIC suffers from almost the same issues as TCP.

A. Peculiarities of high-frequency wireless links

Let us study the peculiarities of high-frequency links focusing on an example of a 5G mmWave system and discussing possible differences between THz and LW links.

We consider a typical 5G deployment scenario illustrated in Fig. 3(a). In this scenario, the network operator uses two frequency bands: (i) a traditional low-frequency band, e.g., a 100 MHz channel at 3.6 GHz carrier frequency, and (ii) a mmWave band, e.g., a channel with an aggregate bandwidth of 1 GHz at 28 GHz. The low-frequency band provides continuous coverage while the mmWave band boosts network performance in hotspot areas. We consider a User Equipment (UE) moving parallel to the line of one low-frequency and two mmWave base stations (called gNBs) with a constant speed of 10 m/s. Starting from 15s till 25s, an obstacle blocks the line of sight between the UE and the first mmWave gNB.

To model this and other scenarios considered in the paper, we use NS-3 [50], which is a popular open-source network simulator actively developed by the research community. NS-3 enables flexible customization of every layer of the OSI protocol stack, as well as implementing novel methods and solutions. Compared to other simulators (e.g., OMNeT++, OPNET, NetSim), NS-3 is widely used for modeling TCP & QUIC performance in wireless systems because it contains the detailed and validated models of: (i) various Radio Access Technologies (RATs) including those operating in mmWave band [51], (ii) various TCP & QUIC congestion control algorithms, (iii) various applications (e.g., web, video, file download). In the numerical results presented in the paper, we show throughput, latency, and other indicators measured at the application layer, which show the overall performance of the whole protocol stack, i.e., including effects arising at the transport layer, link/physical layer of a wireless network, and, if enabled, the effects of cross-layer solutions.

In Fig. 3(b), we show the capacities of the mmWave and the low-frequency links as the functions of time obtained with NS-3. First, we can see that the maximum capacity of the mmWave link (achieved when the UE is close to the corresponding gNB) is approximately ten times higher than the capacity of a low-frequency link because of the ten times higher channel bandwidth. However, the capacity of the mmWave link significantly degrades with the distance, which is caused by the much higher attenuation of a mmWave signal compared to a low-frequency signal. Thus, the coverage of mmWave gNB is limited by several hundreds of meters [13], [52]. As for higher frequency bands (i.e., THz and LW bands), they will potentially contain ten times wider channels than mmWave ones: hundreds of GHz for the THz band vs. tens of GHz for the mmWave band. However, because of much higher path loss and lower power spectral density, which result in low Signal to Noise Ratio (SNR), THz and LW links capacities are expected to be only several times higher than mmWave one with the coverage of dozens of meters [8], [53]–[55].

Second, Fig. 3(b) shows that an obstacle (e.g., buildings, cars, people, trees) between the first mmWave gNB and the UE significantly reduces the capacity of the mmWave link. This problem is caused by a short wavelength of the mmWave signal. When the wavelength becomes smaller than the size of an obstacle, the waves almost do not bend around the obstacle, and propagation becomes more optic-like [56]. Several studies [9], [10], [57], [58] demonstrate that mmWave penetration loss can reach dozens of dB and, thus, significantly decrease the link capacity. When the Line of Sight (LOS) is blocked (at 15s of the experiment), the gNB can still communicate with the UE using reflections from other objects, i.e., using a Non-Line of Sight (NLOS) path. However, SNR on the NLOS path is much lower than that for the LOS path, which drops the link capacity. The link capacity jumps up at 25s when the obstacle disappears from the LOS path and the gNB switches to the beam following the LOS path. The described above effect, i.e., frequent LOS/NLOS transitions, leads to significant fluctuation of the mmWave link capacity. Note

that the LOS/NLOS period is a random value that depends on the environment. The LOS/NLOS fluctuation problem is exacerbated in THz and LW bands, where an obstacle typically causes a link blockage/outage, i.e., almost zero capacity [53].

To conclude, the usage of high-frequency bands brings very high capacity (up to dozens or even hundreds of Gbps) on the one side but, on the other side, results in a huge and almost unpredictable fluctuation of the capacity, which, as shown in the following sections, degrades TCP & QUIC performance.

B. Slow Start problem

After a connection setup or after a period of silence (i.e., no data transmission during RTO), TCP and QUIC use the Slow Start algorithm to probe the network capacity and increase CWND [37]. Since the sender does not know the network capacity at the beginning of the connection, it starts with a relatively low CWND $W = W_{IW}$ and increases it by one Maximum Segment Size (MSS) each time it receives an ACK. MSS is the maximum data packet size that can be sent without fragmentation by the intermediate network nodes. The described above algorithm effectively doubles CWND each RTT, i.e., the growth of CWND with time is exponential.

In [37], the IETF community has agreed to set $W_{IW} = 10 \cdot MSS$, where MSS is limited by 1460 bytes (i.e., MSS that fits the maximum transmission unit of Ethernet links). Such choice of W_{IW} was justified by a requirement of supporting a wide variety of Internet links having capacity from several hundreds of Kbps up to several Gbps.

Although CWND grows exponentially with time during the Slow Start algorithm, reaching BDP may take a sufficiently long time for network paths/links with high capacity and RTT. They are called high-BDP paths/links. For example, if $C = 10$ Gbps and $RTT = 30$ ms, it takes around $t_{BDP} = RTT \log_2(C \cdot RTT / W_{IW}) = 0.34$ s to reach BDP. During this time interval, the network path is underutilized. This problem is called the Slow Start problem [59].

High-frequency links are exactly high-BDP links that suffer from the Slow Start problem. Let us use existing analytical models (e.g., from [60], [61]) to evaluate how the Slow Start problem influences TCP & QUIC performance depending on the wireless link capacity C , RTT between the server and the UE, and the amount of data (file size) S the server sends to the UE. In the model, we assume that after reaching BDP, CWND does not reduce. Thus, the file is transmitted with a rate limited by the wireless link capacity C .

Figure 4 shows the file download time t_{DL} (including the time needed to set up a connection) and the channel resource utilization η that is defined as the ratio of the actual file download rate S/t_{DL} and the link capacity C : $\eta = \frac{S}{C \cdot t_{DL}}$. We can see that when the wireless link capacity $C < 1$ Gbps, which corresponds to the existing 4G/Wi-Fi systems, the Slow Start problem does not influence TCP & QUIC performance: (i) the file download time is inversely proportional to C , i.e., $t_{DL} = S/C$ (ii) the channel resource utilization is close to one. However, when the link capacity exceeds 10 Gbps, typical for mmWave, THz, and LW links, we can see that: (i) the channel

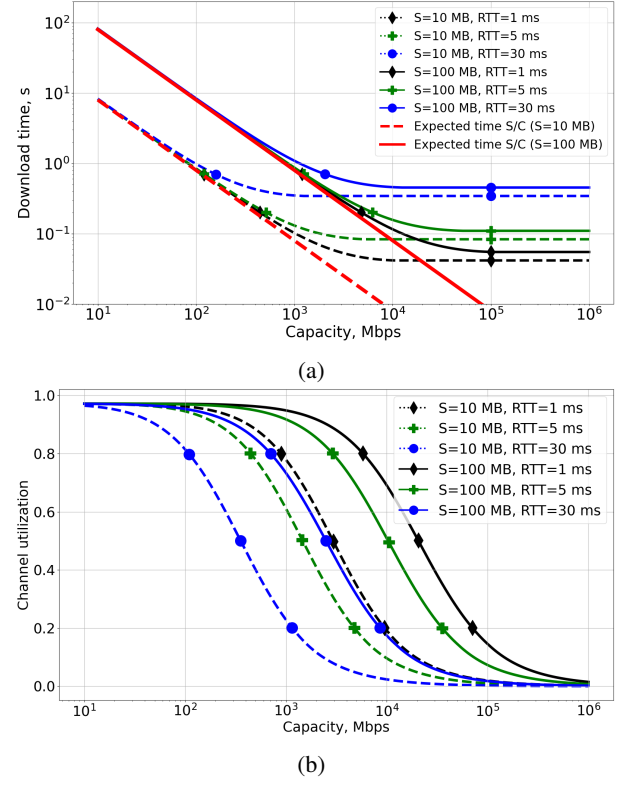


Fig. 4: Slow Start problem: (a) file download time and (b) channel resource utilization as the functions of the channel capacity, overall path RTT, and file size.

becomes underutilized if the transmitted files are small, and (ii) the download time reaches a plateau. The reason for such behavior is that for very high values of BDP (i.e., higher than the file size), the sender executes the Slow Start algorithm during which it sends small portions of data (i.e., limited by small CWND) each RTT. Since the data-sending rate (i.e., the ratio of CWND and RTT) is much smaller than the link capacity, we can see significant link underutilization. Besides that, we can see that the time need to deliver a file t_{DL} mainly depends on RTT rather than on the link capacity.

We can conclude that in contrast to the existing wireless technologies providing channel capacities up to 1 Gbps, ultra-high bandwidth wireless links suffer from the Slow Start problem (especially for high RTT values) that leads to significant underutilization of the available channel capacity.

C. Channel fluctuation problem

Consider a scenario with the transmission of a bulk flow (e.g., ultra-high-definition or 3D video flow) over a mmWave link when the channel periodically changes from LOS to NLOS state and back (e.g., an obstacle periodically appears between the transmitter and the receiver). In addition to the bulk flow, we also run a ping application that rarely sends small UDP packets to measure packet transmission delay.

In Fig. 5, we consider the case when LOS to NLOS transition period is equal to 20 s, i.e., from 0 to 20 s the channel state is LOS, from 20 s to 40 s the channel state is

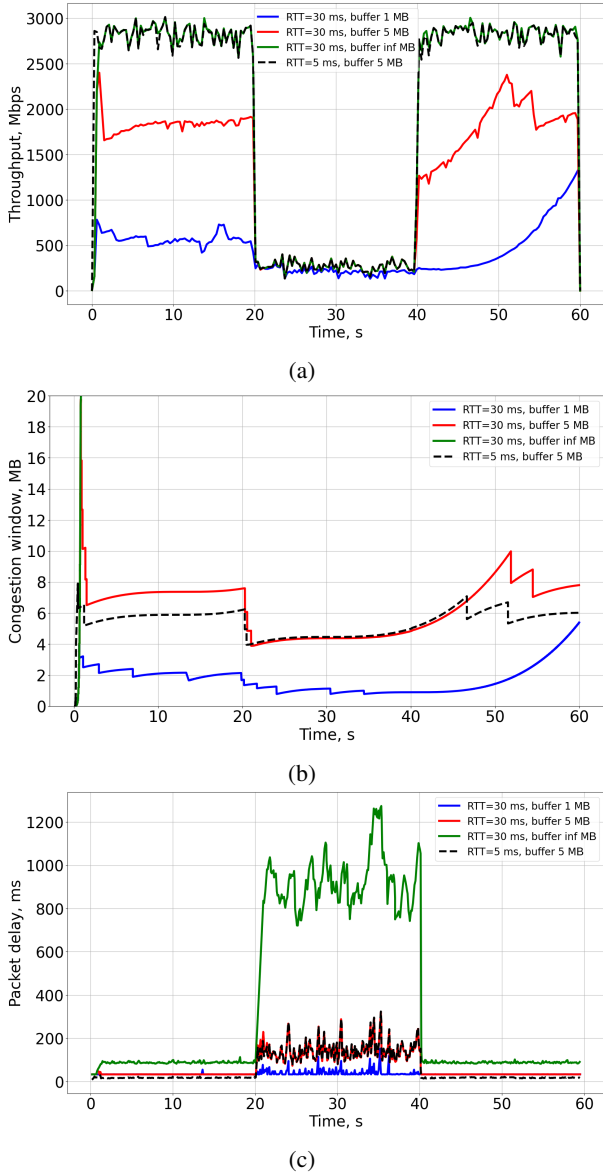


Fig. 5: Illustration of the channel fluctuation problem in mmWave communications: (a) TCP throughput, (b) CWND, (c) one-way delay of UDP packets.

NLOS, etc. We plot: (a) the throughput of a TCP connection, i.e., the average amount of data delivered in a time unit, (b) CWND at the sender, and (c) the one-way delay experienced by UDP packets. In the experiment, we vary RTT between the server and the gNB and the gNB buffer size. The case “infinite buffer” at the gNB is achieved by setting RWND at the UE to 30 MBs and the gNB buffer size greater than RWND. In such a case, the buffer at the gNB does not overflow (i.e., no packets are lost). Thus, CWND increases to infinity (see the corresponding curve in Fig. 5(b)). However, as mentioned in Section II-A, the data sending rate is limited by the minimum of RWND and CWND.

From 0s to approximately 1s, we can see an exponential increase in CWND because the sender executes the Slow

Start algorithm after the TCP connection setup. When CWND becomes greater than the current BDP value plus gNB buffer size, several packets are dropped at the gNB. When the sender detects losses it transits to the Fast Retransmit and Fast Recovery algorithms during which it reduces CWND several times (to make it lower than BDP) and retransmits lost packets. At approximately 2s, all lost packets are recovered and the sender starts executing the CA algorithm.

In this experiment, we consider a CA algorithm called CUBIC, which is the default CA algorithm in Linux-based OSs and iOS [38]. The key idea of CUBIC is to gradually increase CWND with time according to a cubic function. The parameters of this function (i.e., the cubic function inflection point and shape) are selected so that CWND is close to the current BDP estimation at the sender. Specifically, the parameters of a cubic function are chosen as follows. When a congestion event occurs, the sender: (i) reduces CWND: $W = \beta \cdot FlightSize$, where $FlightSize$ is the current amount of sent and unacknowledged data and $\beta = 0.7$ is the fixed algorithm parameter, (ii) saves $FlightSize$ as the current BDP estimation, (iii) starts Fast Retransmit and Fast Recovery algorithm. After Fast Recovery, the CA algorithm increases CWND according to a new cubic function for which the inflection point is selected based on the previous BDP estimation.

Consider what happens when the LOS to NLOS transition occurs at 20 s. The channel capacity instantly drops approximately ten times. The sender tries to reduce CWND several times to react to the change in BDP, but it cannot do it instantly. Thus, we can see a significant increase in packet delays (see Fig. 5(c)). This problem is known in the literature as bufferbloat [62]. As discussed in [62], developers of wireless technologies prefer to keep long buffers at wireless links. On the one side, long buffers are beneficial for achieving high link throughput. But, on the other side, they significantly increase packet delays (up to several seconds), which is harmful to real-time and interactive applications that share the same buffer with a bulk TCP or QUIC flow.

When the channel switches back from NLOS to LOS at 40 s, the channel capacity instantly increases. The CA algorithm monotonically increases CWND according to a cubic function. However, the CWND growth is very slow because of outdated BDP estimation. Only after 50 s, the cubic function changes its slope and starts increasing much faster. Figure 5(a) shows that in the case of a low buffer at the gNB, the NLOS to LOS transition leads to a significant channel underutilization. In contrast, a high buffer at the gNB allows instantly filling up the link and utilizing the available channel capacity. However, it leads to the bufferbloat problem during the NLOS state.

From the considered example, we see that high channel fluctuation inherent to mmWave links can lead to the following problems: (i) bufferbloat when the link changes from LOS to NLOS state, (ii) link capacity underutilization when the link changes from NLOS to LOS state. Similar problems are reported in many papers considering other existing CA algorithms [63]–[65]. The main reason is that the existing CA algorithms try to smoothly change CWND and cannot instantly

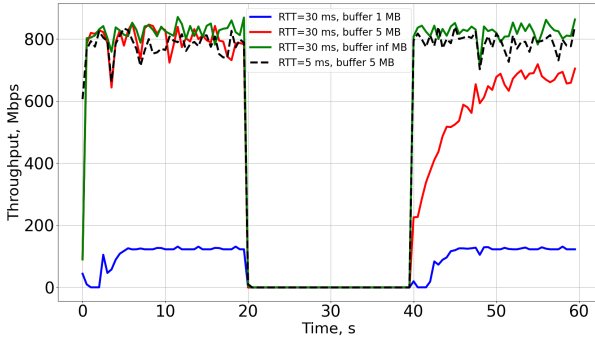


Fig. 6: Illustration of the channel fluctuation problem in Li-Fi (IEEE 802.11bb): TCP connection throughput.

react to channel capacity changes, leading to overestimating or underestimating the required CWND.

Notably, switching from mmWave to THz and LW only aggravates this problem because these frequencies have worse propagation properties and even a thin object can block the signal. Figure 6 shows TCP performance over a single LW link between a Wi-Fi access point mounted on the ceil and a laptop. When a user closes the LW receiver with his arm (between 20s and 40s), the throughput drops to zero and then slowly recovers. Replacing TCP with QUIC does not solve the problem and the curves are similar to those shown in Fig. 5–6.

The paper considers two main directions to solve the problems mentioned above. The first one includes novel CA algorithms, which more accurately estimate BDP or predict LOS/NLOS transitions (see Section VI-A). The second one is Active Queue Management (AQM) algorithms, which are deployed at the gNB/UE and control the amount of the enqueued data so that the wireless link is fully utilized while limiting packet delay (see Section VI-B).

D. Link blockage problem

As mentioned in Section III-A, high-frequency wireless links are subject to high outage probability and have very low coverage. This problem is especially crucial for future THz and LW systems, where an obstacle appearing in the LOS path typically leads to link blockage.

This problem can be solved in different ways. The straightforward approach is to boost the density of gNBs to increase the probability that at least one strong LOS path exists between each UE and some gNB [53]. However, this approach requires high capital expenditures and operational expenses (CAPEX/OPEX).

In recent years, Reconfigurable Intelligent Surfaces (RIS) [66], [67] received much attention as a possible direction to improve coverage and the quality of high-frequency communications. In contrast to usual surfaces, e.g., walls that scatter and attenuate the signal, RIS can focus the signal to the intended receiver, thus, significantly increasing the signal strength of the corresponding NLOS path. Being technology-agnostic, much cheaper, and easier to deploy than gNBs, RIS is considered a promising alternative approach to highly dense

	Below Transport Layer		Transport Layer
Network Architecture	RTT reduction (Section V-A)		Transport-Layer Performance Enhancing Proxies (Section V-B)
Traffic Control	Active Queue Management (Section VI-B)	Cross-layer Solutions (Section VI-C)	Congestion Control Algorithms (Section VI-A)
Multipath Communications	Multi-connectivity (Section VII-A)		Multipath TCP/QUIC (Section VII-B)

Fig. 7: Taxonomy of the solutions boosting TCP & QUIC performance in high-frequency wireless networks.

gNBs. However, many issues are still open, e.g., how to implement low-complexity calibration and instant re-configuration of the RIS to each ongoing transmission.

While RIS is under investigation, currently, the common approach is to use additional low-frequency links that are more stable to environment changes (we consider other ways in Section VII-A). In particular, the 5G specifications allow a UE to connect to several gNBs. This feature is called dual/multi-connectivity [68], [69] which, in theory, allows achieving the throughput equal to the sum of the throughputs of the corresponding wireless links. For example, in Fig. 3, when the UE is in the NLOS state, a part of its traffic can be forwarded to the low-frequency link, which increases the total throughput. In case of link blockage, the low-frequency link allows retaining connectivity but with a much lower data rate.

From the transport-layer perspective, multi-connectivity can be used in two ways. The first way is to use multi-path extensions of TCP [70] or QUIC [71]. These extensions allow the TCP & QUIC sender to set up and control several physical data paths. In such a case, the sender itself determines which fraction of traffic is sent over each data path. The second way is to hide from the TCP & QUIC sender that several physical data paths exist and manage traffic balancing at the link layer. In this case, several gNBs cooperatively manage data packet transmission while UE makes packet ordering and duplicate removal. As a result, the TCP & QUIC sender sees a single aggregated data path with specific properties.

Both described above ways have a common fundamental problem of traffic balancing between several data paths with different capacities and RTT. To address this problem, in this paper, we analyze in detail the existing solutions aimed to improve TCP & QUIC performance in case of frequent switching between high-frequency and low-frequency links.

IV. SOLUTIONS TAXONOMY

During the last three decades, IETF, wireless communications standardization bodies (i.e., 3GPP and IEEE), and the research community proposed a vast number of solutions improving TCP & QUIC performance in both wired and wireless systems. The main goal of this survey is to identify the key ideas behind the existing solutions and analyze whether

they can be directly applied or extended to boost TCP and QUIC performance in high-frequency wireless networks.

We propose a two-dimensional classification for the considered solutions, see Fig. 7. The first dimension covers approaches used to improve TCP and QUIC performance. The second dimension defines the layer of the OSI protocol stack where modifications are performed.

In the first dimension, the first approach changes the *network architecture* (see Section V). For example, some extra nodes can be deployed on a path between a server and a client, or the path properties (e.g., RTT) can be changed. The second approach includes various solutions that *control the amount of traffic* transmitted between the client and the server such that the available bandwidth is fully used without inducing congestion/bufferbloat at the intermediate nodes (see Section VI). The third approach uses *multipath communications* in modern wireless systems when the client is connected to the network via several wireless channels (see Section VII).

The second dimension for the classification is the layer of the OSI protocol stack at which a particular solution is implemented. Specifically, a vast number of solutions work at the transport layer, i.e., these solutions extend or modify the TCP or QUIC protocols or the algorithms implemented at a TCP/QUIC sender. The other group of solutions works below the transport layer. Such solutions indirectly influence TCP/QUIC flow (e.g., by prioritizing or dropping packets at intermediate nodes) in order to improve the overall performance. Also, we consider cross-layer solutions that benefit from interaction between the transport layer and other layers (e.g., the link layer of a base station).

In the following sections, we consider in detail each group of solutions and analyze whether they can address three problems identified in Section III. Since TCP is a more mature and widespread transport layer protocol, most of the solutions were designed for TCP. We analyze them in detail and determine which solutions can be directly applied or extended for QUIC. In Table II, we summarize the benefits of the considered solutions, discuss their implementation complexity, determine which of the three problems they can address, and list the band-related issues.

V. NETWORK ARCHITECTURE

This group of solutions modifies network architecture to speed up transport-layer protocols. Specifically, in Section V-A, we consider solutions that reduce the RTT of a path between a server and a client. Lower RTT allows a TCP/QUIC sender to establish a connection more quickly and promptly react to the changes in the network path properties. In Section V-B, we consider Performance Enhancing Proxies (PEP). PEPs are special network nodes deployed on the path between a server and a client aimed to improve the performance of a TCP/QUIC flow.

A. RTT reduction

One of the key factors influencing TCP & QUIC performance is RTT between a server and a user. A lower

RTT speeds up the control loop, i.e., it allows the server to quickly increase CWND during the Slow Start and Congestion Avoidance phases and promptly react to congestion events.

As shown in Fig.3(a), in cellular systems, the whole RTT between a server and a user consists of three main components: (i) RTT_{S-PGW} is the RTT between the server located on the Internet and the Packet Gateway (PGW), i.e., the gateway of the cellular system to the Internet, (ii) $RTT_{PGW-gNB}$ is the RTT inside the core network, i.e., between the PGW and the serving gNB, (iii) RTT_{gNB-UE} is the RTT of the wireless link between the gNB and the UE. Below we describe solutions to reduce each component of the RTT.

1) Solutions to reduce RTT:

a) *RTT between server and PGW*: The RTT between a server and a PGW mainly depends on the geographic locations of these nodes. For example, the RTT between a server located in the USA and a PGW in Europe may exceed 100 ms. To reduce RTT, many content/service providers employ Content Delivery Networks (CDNs). The main idea of CDN is to deploy a distributed network consisting of many servers (called replica servers) that are located as close as possible to end users. Servers periodically synchronize their content (web pages, video files, etc.) to keep them up-to-date. When a UE wishes to download some content, it first finds the address of the closest replica server, e.g., by using the Domain Name System (DNS) query. Then, the UE establishes a TCP or QUIC connection with the found replica server.

The key problem in designing CDNs is determining how many and where replica servers should be placed to reduce RTT with the lowest increase in CAPEX/OPEX. During the last two decades, much research has been done in this area. We refer the interested reader to a recent survey on this topic [72].

A network operator can further reduce RTT by caching the most popular data inside the network (e.g., on caching routers) [118]–[120]. When a router sees a HyperText Transfer Protocol (HTTP) request of some content cached inside the network, the request can be redirected to the corresponding node. Caching routers typically have considerably smaller cache sizes compared to replica servers. However, the most popular data can be located close to the user than the replica server. The major drawback of caching routers is that they assume that the application layer data (e.g., HTTP request) can be accessed at the routers, which is not possible with modern end-to-end encryption protocols (e.g., TLS).

Content routers [121], an advanced version of caching routers, mainly identify packets by content characteristics (e.g., application header fields, content service name) instead of IP addresses to categorize and prioritize content types. In [122], authors propose a caching method that estimates content popularity based on data request statistics and uses this information for cache management. In [123], authors develop a content caching design for 5G systems that dynamically controls caching decisions to maximize the average requested data rate without using the knowledge of content popularity.

At the early stage of CDN development, each service provider had to deploy and manage its physical servers or rent

TABLE II: Summary of the solutions aimed to improve TCP & QUIC performance in high-frequency wireless networks.

#	Solution Type	Modification/Subtype	Implementation	Applicability to QUIC	Slow Start	Channel fluctuation	Link blockage	Benefit	Issues related to mmWave, THz, and LW bands
1	RTT reduction	Between server and wireless gateway [72]–[74]	Content/service provider deploys additional CDN replica servers	Applicable	+	+	-	Lower RTT improves congestion control algorithms convergence, thus, increasing data download/upload rate and reducing queueing delays	RTT reduction does not depend on the used band
		Inside core network [75]	Network operator deploys (virtual) PGWs close to base stations	Applicable	+	+	-		Increasing frequency from mmWave to LW shortens slots and reduces the wireless link RTT
		On a wireless link [3], [76]	Network operator uses/deploys modern RATs that reduce data delivery time	Applicable	+	+	-		
2	Transport-Layer Performance Enhancing Proxies	Split connection PEP [77]–[79]	A TCP flow shall be served by the same PEP during the link degradation/blockage	Not applicable because of QUIC header encryption	+	+	-	PEPs deployed inside the core network can obtain information about wireless links characteristics and shape TCP and QUIC flow accordingly	As moving obstacles may block THz or LW links, the PEP shall be deployed in the core network and connected with the base stations operating in various bands
		Transparent PEP [80]–[82]		Only PEPs that do not use information from QUIC header	-	+	-		Implementation is independent of the used band. However, the hardly predictable quality of THz and LW links requires the design of new decision-making algorithms.
3	Congestion Control Algorithms	Slow Start Modifications [83]–[86]	Implemented at endpoint devices: servers (for DL traffic) and mobile devices (for UL traffic)	Applicable	+	-	-	Different CA algorithms aim to find a tradeoff between throughput and latency. However, there is no single algorithm that provides Pareto improvement in all scenarios	Existing Slow Start and CA algorithms are too slow to quickly adapt CWND for highly fluctuating BDP in THz/LW bands. Inherent to THz and LW long link blockage may induce multiple RTOs and significantly degrades the performance.
		Congestion Avoidance Algorithms [30], [33], [87]		Applicable	-	+	-		
4	Active Queue Management (AQM)	Single-queue AQM [88]–[95]	Implemented at the link layer of base stations and/or mobile devices	Applicable if QUIC flows are treated at the link layer as congestion responsive flows	-	+	-	Provide limited queueing delay. However, the throughput of a bulk flow can degrade due to rapid wireless link capacity degradation	Required buffer size grows when switching from mmWave to THz and LW bands, which increases the complexity and cost of base stations
		Multi-queue AQM [96]–[103]			-	+	-	Provide both high throughput for bulk flows and limited queueing delay for delay-sensitive flows	
5	Cross-layer solutions	In-device cross-layer solutions [104]–[106]	Endpoint devices can exchange the cross-layer information between the link layer and the transport layer protocols	Applicable	+	+	-	Control information (e.g., the capacity of a wireless link) can assist CWND selection at endpoint devices. Proper CWND selection allows efficiently utilizing the available wireless link capacity and reducing latency	Cross-layer solutions increase the complexity of base stations. Thus, lightweight versions should be designed for low-cost THz/LW base stations. As the frequency grows, the wireless link capacity becomes less predictable, which requires new algorithms for transmission parameters (e.g., CWND) selection at endpoints.
		TCP/IP header modification [107], [108]	Endpoint devices and wireless network devices (e.g., base stations) exchange cross-layer information via TCP/IP headers	Partially, only IP header fields can be modified (e.g., the ECN bit)	+	+	-		
		Special service protocols [28], [109], [110]	Endpoint devices and the core network devices shall support common service protocols	Applicable	+	+	-		
6	Multi-connectivity	Algorithms balancing traffic between several wireless links [69], [111], [112]	Implemented at the link layer of base stations and mobile devices. Implementation depends on the used RAT	Applicable	-	+	+	Provide higher total throughput by utilizing resources of several wireless links. Enable fast traffic re-routing in case of link outage	Advanced MC features (e.g., COMP) are hardly possible in LW and THz bands because of too high complexity unaffordable to cheap THz/LW devices. It is worth extending 3GPP Multi-radio Dual Connectivity and Wi-Fi Multi-link for THz/LW bands.
7	Multi-path TCP/QUIC	Congestion Control Algorithms for the multi-path case [113]–[117]	Implemented at endpoint devices: both servers and mobile devices	Applicable	-	+	+	Provide higher total throughput by utilizing resources of several data paths (e.g., a mobile device can be connected to different RATs). Long traffic re-routing in case of path congestion/blockage	The performance of MPTCP/MPQUIC significantly degrades if a link is frequently blocked (e.g., in THz/LW bands).

them from other third-party companies (e.g., Akamai, Amazon) and/or large telco operators. The emergence of the Network Functions Virtualization (NFV) paradigm standardized by ETSI has led to the appearance of so-called Cloud-based CDN [72]. According to the NFV paradigm, various server functions (storage, image/video rendering) are implemented in software and potentially can be executed on any type of hardware (e.g., even at a gNB having enough computational and storage resources). NFV allows the service/CDN provider to dynamically configure and place virtual functions based on current user demands without the need to deploy or rearrange existing hardware infrastructure.

A complimentary to the CDN paradigm that emerged in recent years is Multi-access/Mobile Edge Computing (MEC) [73], [74]. Many existing and future mobile applications (e.g., cloud gaming, cloud augmented and virtual reality, intelligent transportation [124], [125], etc.) require a high amount of computing resources and cannot be executed on a mobile device. For this reason, a computation task should be sent (in particular, by using TCP or QUIC) and executed at some remote server. Besides computing resources, such applications typically require very low latency to execute a computation task (i.e., low RTT). To provide a low RTT, MEC also uses NFV and allows the cellular operator to place virtual functions close to the radio access network (i.e., at gNBs or close to them).

To improve the network performance for HTTP traffic, the network operator can deploy HTTP proxies [126]. HTTP proxy is an intermediate server inside the operator core network that either filters and redirects the HTTP request to the original server or sends the data to the client itself if the required data are found in the cache. Besides that, with an HTTP proxy, two separate TCP connections are established: (i) between the client and the proxy, and (ii) between the proxy and the server. Similar to TCP split connection PEP (see Section V-B), it allows speeding up Slow Start because each TCP connection has a lower RTT. HTTP proxy can be easily implemented since it only requires configuration of the software (e.g., browser) at users. However, due to additional proceeding delays at HTTP proxies, the performance in some cases can even degrade.

b) RTT between PGW and gNB: If the content or a virtual function is placed on a third-party server (i.e., the server that does not belong to the operator's core network), the corresponding data flow shall traverse PGW. In 4G systems, a PGW is a separate network node typically executed on specialized hardware. Network operators deploy few PGWs taking into account geographical and CAPEX/OPEX constraints. Thus, the RTT between the gNB and PGW $RTT_{PGW-gNB}$ can be comparable to RTT_{S-PGW} . In contrast, the novel 5G core network architecture employs the NFV paradigm according to which PGW functions can be considered virtual network functions and executed on any appropriate hardware [75]. Thus, if PGW functions are executed at the gNB or close to it, we can achieve almost zero $RTT_{PGW-gNB}$.

c) RTT between gNB and UE: This component of RTT depends on the used RAT and its characteristics. In cellular systems, the time axis is divided into slots. The base station schedules data transmission in each slot. In 4G systems, the slot duration is fixed and equals 1 ms. However, the transmitter and receiver require additional time to schedule, encode and decode data, which takes approximately four slots. In the case of transmission failure, every additional retry takes at least eight slots. Thus, in 4G systems, RTT_{gNB-UE} can reach several dozens of milliseconds, which is comparable to RTT_{S-PGW} and RTT_{S-PGW} .

The 5G specifications [3] introduce a novel RAT called New Radio (NR). In NR, the slot duration can be selected flexibly, depending on the served traffic. In particular, the authors of [76] propose to use short slots of hundreds of microseconds during the Slow Start phase and demonstrate that short slots significantly increase the data download rate.

2) Applicability to QUIC: The described above solutions do not depend on the considered transport layer protocol and, therefore, they improve the performance of both TCP and QUIC. Note that QUIC establishes connections faster than TCP (see Section II-B). Thus, QUIC reduces the overall transmission time. However, this effect is notable only for the case of very short data sizes (i.e., dozens of kB corresponding to the default value of initial CWND) and high RTT [18].

3) Benefits of lower RTT and impact of various bands: Let us analyze how short RTT improves TCP & QUIC performance. In Fig. 4, we consider three reference RTT values: (i) 30 ms that corresponds to a remote server, (ii) 5 ms that corresponds to a local server or the usage of CDN, (iii) 1 ms that corresponds to the MEC or Cloud CDN approaches when the server is located near the gNB. We can see that low RTT values partially solve the Slow Start problem, i.e., it increases resource utilization and reduces file download time. Moreover, many papers (e.g., [24], [63], [127]) show that low RTT speeds up the control loop of CA algorithms. Thus, they promptly react to changes in link capacity and properly select the CWND (see the results presented in Fig. 5–6).

The usage of high-frequency bands may additionally speed up the control loop by further reducing RTT over a wireless link as follows. The higher is the frequency, the wider are the used channels and the smaller is the transmission range. Consequently, the wireless devices can use higher order numerology, i.e., they can shorten the slots well beyond hundreds of microseconds mentioned in Paragraph V-A1c. Thus, the usage of THz and LW bands may additionally accelerate the TCP & QUIC control loop and improve their performance.

B. Transport-Layer Performance Enhancing Proxies

TCP was initially developed for wired networks in which the data path between a server and a user is composed of links with almost constant characteristics: capacity and transmission delay. However, the rapid development of wireless technologies, both satellite and terrestrial ones, has led to a situation when some links (typically, last-hop links) have unique characteristics. For example, the earlier RATs did not

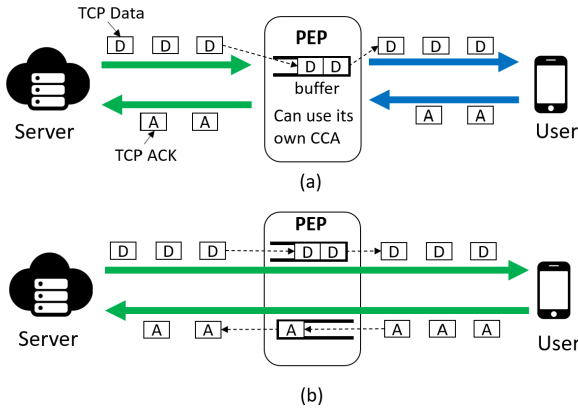


Fig. 8: Performance Enhancing Proxy (PEP) architecture: (a) split connection PEP, (b) transparent PEP.

provide link-layer retransmissions leading to packet losses because of transmission errors. However, TCP treats any packet loss as a congestion signal and tries to reduce CWND and data sending rate. Thus, for high transmission error rates inherent to wireless links, TCP performance significantly degrades [128], [129]. Another example is a satellite link that has a varying capacity and extremely high RTT.

As mentioned in Section II-A, because of transport layer abstraction, TCP cannot take into account the characteristics of each link. To improve the performance of TCP in the case when the data path contains a link with specific characteristics, researchers proposed to employ special entities called Performance Enhancing Proxies (PEPs). PEP is an entity that is deployed at some network node (typically close to links with unique characteristics) to “help” TCP endpoints overcome losses or performance issues caused by these links. In particular, to hide from TCP-endpoint transmission errors, PEP deployed at a wireless link (e.g., at the base station) can retransmit lost packets without waiting for long retransmission from the server [128], [129].

During the last three decades, many PEP architectures have been proposed to improve TCP performance in various environments. IETF describes some of them in [130]. According to [130], PEPs operating at the transport layer⁵ can be classified into two main categories: (i) split connection PEPs, (ii) transparent PEPs.

1) Description of PEPs:

a) *Split connection PEPs*: As the name states, a split connection PEP divides the original TCP connection into two separate TCP connections: (i) from the server to the PEP and (ii) from the PEP to the user (see Fig. 8(a)). In such an architecture, PEP represents an intermediate server that downloads data from the original server using the first TCP connection and then delivers the data to the user using the second TCP connection. Note that even with the split connection PEP, it is possible to hide from endpoints that the connection is split into two or more parts by properly

modifying TCP header fields (IP addresses and ports). With this scheme, the endpoints consider that they obtain TCP packets from the other endpoint.

The benefits of split connection PEPs are as follows. First, RTT on each TCP connection becomes lower than that of the original TCP connection. Specifically, PEP can send TCP ACKs on behalf of the user without waiting for the original TCP ACKs from the user. However, by doing such a trick, PEP becomes responsible for delivering the corresponding data packet to the user and obtaining TCP ACK from it. Second, as PEP becomes a TCP sender on a connection between the PEP and the user, it can use any TCP congestion control algorithm and set any value of the TCP internal parameters. For example, a wireless network operator deploying a split connection PEP can adjust the initial CWND.

Many studies show that split connection PEPs improve TCP performance in cellular [77] (specifically, in 5G [78]) and Wi-Fi networks [79]. Specifically, if the PEP uses a high initial CWND, it can increase the file download rate by accelerating Slow Start. The gain is especially high when the RTT between the PEP and the server is lower than that between the PEP and the user [78]. Because of a short RTT between the PEP and the user, existing CA algorithms have a faster control loop to adapt CWND to varying wireless link characteristics [24]. Moreover, a split connection PEP installed close to wireless links (e.g., at the gNB) can use its own CA algorithms. In [79], the authors propose selecting CWND based on BDP estimation measured at the mmWave link and showing that it allows efficiently using fast varying mmWave link capacity.

The main drawback of split connection PEPs is that they break the end-to-end semantics of TCP. When the PEP sends a TCP ACK on behalf of the user, the server assumes that the data are delivered to the user and removes it from its send buffer. However, PEP cannot ensure further delivery of this packet to the user (e.g., the user can disconnect from the wireless network). Developers advocating the usage of split connection PEPs state that applications using reliable TCP transport should not rely on TCP ACKs to ensure successful data delivery [130]. Instead, the server should rely on specific application layer acknowledgments. For example, when File Transfer Protocol finishes the transmission of a file, the receiver sends a short message with the command indicating successful file delivery.

b) *Transparent PEPs*: In contrast to split connection PEPs, transparent PEPs retain end-to-end semantics. Transparent PEPs are deployed on the path between a server and a user and can affect TCP data flows as follows: (i) they can buffer/delay TCP packets, both data and ACKs (see Fig. 8(b)), (ii) they can retransmit some TCP packets without waiting for the retransmission from the original server, (iii) they can modify some TCP header fields (e.g., the size of RWND). Transparent PEPs are typically deployed in the operator’s core network and use the knowledge of the characteristics of the wireless link to improve data delivery.

Many papers propose transparent PEPs to improve TCP performance in 5G mmWave networks. In [80], [81], the authors

⁵PEPs operating at other layers (e.g., HTTP proxy, caching routers) that aim to reduce effective RTT are considered in Section V-A.

propose a mmWave PEP (mmPEP) that buffers data packets during the NLOS channel state and dispatches them when the channel switches to the LOS state. Also, mmPEP retransmits packets if some of them were dropped at the gNB because of buffer overflow or link outage. The authors of [80], [81] demonstrate that mmPEP alleviates the bufferbloat problem by reducing sending rate during the NLOS state and increases channel utilization by increasing sending rate during the LOS state. A similar idea is considered in [82] in which PEP buffers data packets and dispatches them with the rate corresponding to the current BDP estimation. To avoid PEP buffer overflow, the authors of [82] recommend changing the RWND field in TCP ACKs that are sent back to the server. Thus the receiver makes the server reduce the rate with which it injects packets into the network. Managing RWND at PEP to avoid bufferbloat problems is also considered in [131], [132].

2) *Applicability to QUIC*: As detailed in Section II-B, the QUIC header is encrypted, therefore, intermediate networks nodes (e.g., PEPs) cannot: (i) split QUIC connection and (ii) modify QUIC header. So, a very limited set of PEP-based solutions can be adapted for QUIC. For example, the described above mmPEP solution that buffers and dispatches packets can be used. However, since PEP cannot reveal ACK sequence numbers encrypted in QUIC packets, fast retransmission of lost data packets is not possible.

3) *Implementation of PEPs with various bands*: In theory, PEPs can be deployed at any intermediate node including base stations, and used with any high-frequency band. However, in practice, it is better to locate PEPs in the core network, rather than at the base stations, especially in the case of Split Connection PEP.

When a moving obstacle completely blocks the high-frequency link between a user device and a base station – which is especially relevant to THz and LW bands — the connection needs to be (re)established via another base station operating in the same band or even in a low-frequency one. Thus, sending the data to the PEP located at the old base station and then to the new one results in an extensive load of the backhaul links between the core network and the base stations. Note that, as the coverage reduces for higher frequencies, not all mmWave base stations, to say nothing about THz and LW, are collocated with the low-frequency ones. Besides that, the computational power of THz and LW base stations is limited, which also prevents the usage of PEPs at base stations.

Another issue is that prediction of the quality of THz and LW channels and, consequently, their capacity is more complicated than for mmWave [133], which requires the design of new decision-making algorithms for Transparent PEPs.

VI. TRAFFIC CONTROL

Traffic control for TCP/QUIC connection should provide the highest throughput without inducing congestion/bufferbloat at intermediate nodes. This task can be solved at different layers of the protocol stack. First, it is solved at the transport layer with a congestion control algorithm implemented at the

TCP/QUIC sender. In Section VI-A, we review numerous congestion control algorithms proposed so far and determine which of them are promising for high-frequency wireless links. Second, intermediate nodes (e.g., base stations) can shape the TCP/QUIC flow by dropping or prioritizing some packets. Therefore, in Section VI-B, we analyze various Active Queue Management algorithms. Third, in Section VI-C, we consider cross-layer solutions that use the interaction between the transport layer and the lower layers (e.g., the link layer of wireless devices) to improve traffic control.

A. Congestion Control Algorithms

As detailed in Section II-A, the congestion control algorithm is a key algorithm executed at the TCP sender that determines the data sending rate and, therefore, the overall TCP connection performance. For long TCP connections, most of the time, packets are transmitted without losses, and the TCP sender tries to probe available bandwidth using either Slow Start or CA algorithms. In particular, after TCP connection setup or after a major congestion event, i.e., after a retransmission timeout (RTO) expiration, the TCP sender uses the Slow Start algorithm. During the other time, it uses the CA algorithm. During the last decades, researchers and IETF have proposed numerous modifications of the Slow Start algorithm and novel CA algorithms aiming to improve TCP performance for high-BDP links. Some of them became widely spread and are implemented in the existing OSs, while others are still experimental. In this section, we classify these numerous proposals and analyze which of them are promising for improving TCP performance in high-frequency wireless networks.

1) Congestion Control Improvements:

a) *Slow Start modifications*: The rapid increase in access links capacities (both wired and wireless ones) from tens of Mbps to several Gbps has led to a situation when most TCP connections are operating over high-BDP links (i.e., links with BDP of several megabytes). As described in Section III-B, the Slow Start algorithm doubles CWND each RTT. In other words, the TCP sender uses a binary search procedure to find optimal CWND. As shown in many papers [84], [85], [134], in high-BDP links, with a high probability, this binary procedure overshoots the optimal CWND, leading to severe packet loss. Consequently, the TCP sender needs to use a long Fast Recovery procedure or even restart the Slow Start procedure starting from 1 MSS if the RTO timer expires.

To address mentioned above problem, several Slow Start algorithm modifications were proposed. In particular, RFC 3742 [83] describes a Limited Slow Start (LSS) modification. The idea of LSS is to slow down CWND growth when CWND reaches a given threshold $LSSThresh$. By default, $LSSThresh$ is set to 100 MSS. After reaching $LSSThresh$, the TCP sender starts increasing CWND linearly, i.e., by $LSSThresh/2$ on each RTT. The main drawback of LSS is that it uses a fixed value of $LSSThresh$ for all scenarios. In fact, $LSSThresh$ should be adaptively selected depending on the actual bottleneck link BDP that is unknown at the TCP sender.

A more sophisticated solution called Hybrid Start (HyStart) is proposed in [85]. In contrast to LSS, HyStart tries to adaptively determine the time moment when the bottleneck link becomes congested and, thus, exit Slow Start earlier without high packet loss. To detect congestion the TCP sender measures the following variables: (i) the interval between consequently received ACKs (called ACK train length), and (ii) RTT estimation for each packet. When one of the two variables starts increasing quickly, the TCP sender switches from Slow Start to CA algorithm. Simulations and experimental results show that HyStart provides sufficiently low false-positive congestion detection probability. Thus, HyStart often exits Slow Start when CWND reaches BDP without significant packet loss [85]. HyStart is currently implemented and used by default in Linux-based OSs. Recently, HyStart++ modification was proposed that combines HyStart and LSS [86]. Specifically, if HyStart++ detects high RTT growth, the TCP sender switches from Slow Start to LSS.

As the described above solutions aim to avoid massive packet losses during Slow Start, they are very conservative in increasing CWND, e.g., some of them slow down CWND growth when it approaches BDP. Therefore, these solutions do not address the Slow Start problem described in Section III-B.

b) Congestion Avoidance Algorithms: As mentioned in Section II-A, in case of a low packet loss rate, most of the time, the TCP sender is executing the CA algorithm. The goal of the CA algorithm is to smoothly increase CWND to fully use the available bandwidth while reducing CWND when the bottleneck link becomes congested. So far, more than a hundred CA algorithms have been developed to improve TCP performance for different types of links (wired, wireless, satellite) and network scenarios. In this section, we follow a taxonomy of CA algorithms proposed in [30] (other recent taxonomies and analysis of exiting CA algorithms can be found in [33], [87]). We describe the main design principles of different types of algorithms, provide examples of the most widely used algorithms and analyze which of them provide better performance in high-frequency wireless networks.

Loss-based algorithms were proposed in the earlier 1980s following the advent of TCP. As the name states, they increase CWND until the buffer at the bottleneck link overflows, and the TCP sender detects packet losses. In other words, they consider packet loss as an indication of congestion.

RFC 5681 [37] (i.e., the IETF document describing the standard architecture of TCP congestion control) proposes the so-called Reno algorithm as a reference CA algorithm. Reno implements the AIMD principle described in Section II-A: each RTT, Reno increases CWND by one MSS until the first packet is lost. When Reno detects a packet loss, it halves CWND, recovers lost packets (i.e., runs the Fast Recovery procedure), and again starts linearly increasing CWND. Thus, CWND follows a sawtooth-like function of time. In [135], IETF proposed the modification of Reno called NewReno that improves the performance during the Fast Recovery procedure. Currently, NewReno is considered a reference/standard CA algorithm that is implemented in all existing OSs.

With the appearance of high-BDP links, the researchers found that the NewReno linear increase function is too slow to reach BDPs of several megabytes. Therefore, many new loss-based CA algorithms for high-BDP links were proposed, e.g., HighSpeed [136], BIC [137], and CUBIC [38]. As detailed in Section III-C, CUBIC increases CWND as a cubic function of time. The CWND growth function has two regions: (i) flat region where CWND is close to the current BDP estimation and (ii) fast-increasing region where CWND is far from that estimation (see Fig. 5(b)). Because of its simplicity and other useful properties (e.g., RTT-fairness [138]), CUBIC is currently a default CA algorithm used in almost all existing OSs: Linux, iOS, and Windows starting from Windows 10.

Delay-based algorithms use RTT statistics to detect congestion. Specifically, when the bottleneck link becomes congested, its buffer starts growing, which increases the overall RTT. So, in contrast to loss-based algorithms, this type of algorithms can detect congestion earlier and, therefore, reduce or even avoid packet losses.

The simplest example of delay-based algorithms is called Vegas [139]. Vegas keeps a variable RTT_{min} that is the minimal measured RTT (typically, the minimal RTT is observed after TCP connection setup). For the current RTT measurement RTT , Vegas estimates the size of the bottleneck buffer as $\Delta = CWND(1 - RTT_{min}/RTT)$. If Δ is greater than threshold Δ_{high} the bottleneck link is considered to be congested, and CWND is decreased by 1 MSS. If Δ is lower than threshold Δ_{low} , the bottleneck link is underutilized, and CWND is increased by 1 MSS. In other cases, CWND is not changed. During the last decades, several enhancements of Vegas (namely, Vegas-A [140], Vegas-V [141], NewVegas [142]) and more sophisticated delay-based algorithms, such as Verus [143] and Nimbus [144] have been proposed.

Delay-based algorithms have several advantages, such as low delay and low probability of packet loss. However, compared with loss-based algorithms, delay-based algorithms are too conservative: if two TCP flows that use loss-based and delay-based CA algorithms share the same bottleneck link buffer, the loss-based TCP flow will eventually occupy the whole buffer reducing the throughput close to zero for delay-based TCP flows. Since more aggressive loss-based algorithms were designed and implemented historically first, delay-based algorithms did not become widely spread. An interesting exception is Low Extra Delay Background Transport (LEDBAT), specifically developed by IETF [145] for software updates and BitTorrent applications. The traffic of these applications has low priority and can occupy resources based on the left-over principle. So, the described properties of delay-based algorithms exactly fit the requirements of background traffic.

Hybrid algorithms combine the benefits of loss-based and delay-based approaches. A well-known example of hybrid algorithms implemented in Windows is Compound TCP [146]. Compound TCP keeps two congestion windows: (i) a loss-based window that is managed by the Reno algorithm and (ii) a delay-based window that is controlled by a Vegas-like algorithm. The resulting CWND is the sum of the two

windows. When the TCP sender detects a high increase in RTT, it reduces the delay-based CWND component and behaves similarly to Reno. Otherwise, CWND increases faster than Reno, improving the utilization of high-BDP links.

Another example of hybrid algorithms is TCP Illinois [147]. Similar to Reno, it uses the AIMD principle. However, instead of fixed additive increase and multiplicative decrease factors, it varies these factors based on the measured RTT. Other examples of hybrid algorithms include TCP Veno [148], and Yet Another Highspeed TCP (YeAH-TCP) [149]. These algorithms are still experimental in contrast to the mentioned above algorithms, which are implemented in Linux.

Capacity-based algorithms implement an idea to estimate the bottleneck link capacity C and the minimal RTT RTT_{min} . Given these measurements, CWND can be set to the following estimation of BDP: $CWND = C \cdot RTT_{min}$. While RTT_{min} can be easily obtained from RTT measurements, it is difficult to obtain an accurate estimation of C using local measurements available at the TCP sender.

An example widely studied in the literature is Westwood [150]. Westwood estimates C using the series of ACK reception times T_i and the number of bytes B_i acknowledged by each ACK. Since ACK is generated after reception of the corresponding data packet, the capacity sample can be estimated as $C_i = B_i / \Delta T_i$, where $\Delta T_i = T_i - T_{i-1}$ is the interval between two consequent ACKs. To smoothen the high fluctuation of capacity samples, Westwood uses a low-pass filter. While the described above approach is easy to implement, it has several drawbacks that can induce significant errors in estimated bottleneck capacity. First, ACKs that traverse the reverse path can experience additional delays (e.g., processing and queuing delays at intermediate nodes) that bias the distribution of ΔT_i . Second, in wireless systems, several ACKs can be aggregated into one long packet, which results in the so-called ACK compression problem, i.e., almost zero ΔT_i samples. To address these problems, a modification of Westwood called Westwood+ was proposed in [151]. The idea of Westwood+ is to consider the ACK arrival process during a sufficiently long time interval (e.g., during one RTT) to obtain average estimation $C_{avg}(S) = \sum_{i \in S} B_i / \sum_{i \in S} \Delta T_i$, where S is the set of ACKs received during one RTT. While Westwood+ efficiently solves the ACK compression problem, its performance is still influenced by possible congestion at the reverse path, which biases the ACK arrival times and causes wrong estimations of bottleneck link capacity [152].

Another example of capacity-based algorithms recently developed by Google is called Bottleneck Bandwidth and Round-trip propagation time (BBR) [153]. Most of the time, BBR operates in the steady state by setting CWND to the product of measured C and RTT_{min} . To measure C , BBR periodically switches to the ProbeBW state, in which it inflates CWND to probe higher link capacity. In turn, to update RTT_{min} , BBR periodically switches to the ProbeRTT state, in which it significantly reduces CWND to decimate the bottleneck queue.

In recent years, BBR received much attention from academia and industry. Real-life experiments performed by

Google [153] demonstrate that in contrast to loss-based algorithms (e.g., CUBIC), BBR fully uses the available capacity without inducing the bufferbloat problem. However, many studies report open issues among which: (i) high packet loss in the presence of short buffers [154], (ii) unfair behavior to existing loss-based TCP flows [155], [156], (iii) low performance in wireless networks with high user mobility [157], [158]. Note that the third issue is relevant to high-frequency wireless links that intrinsically have fast varying capacity. Some issues were addressed in BBR version 2 [159]. Currently, BBR is under wide area testing in Google services such as YouTube and Google Cloud.

ML-based algorithms use Machine Learning (ML) methods to improve TCP congestion control. Most of the algorithms proposed so far (e.g., QTCP [39], TCP-RL [160], SmartCC [161]) are based on Reinforcement Learning (RL) approach. According to this approach, the TCP sender, called the agent, can make actions (e.g., send packets, or change CWND). In response to these actions, the environment (i.e., the network) provides a reward that can be measured in terms of throughput and/or RTT. The agent tries to find an optimal policy (i.e., a sequence of actions) providing the maximum reward. The key problem in designing efficient RL-based CA algorithms is how to define the set of possible actions and select a proper reward function. Besides that, the algorithm should quickly learn, i.e., adapt the optimal policy for fastly varying network conditions. We refer the interested reader to a recent survey [162] for a detailed analysis of the existing ML-based algorithms.

We should note that the research on ML-based CA algorithms is still in its infancy. While the authors report the advantages of the proposed ML-based algorithms with respect to the legacy rule-based algorithms such as CUBIC and NewReno, their analysis is based on pure simulations and is limited to specific scenarios. Thus, more effort is required to implement and validate novel algorithms in real systems, prove their stability and ensure their fair coexistence with the legacy algorithms.

c) Evaluation of CA algorithms: The research on CA algorithms has been continuing for over thirty years. The appearance of ultra-high capacity mmWave links with the unique properties described in Section III-A has led to a new cycle of research interest in CA algorithms providing both high throughput and low latency over such links. In particular, many papers analyze and compare the performance of the existing CA algorithms in mmWave networks using analytical models [25], [127], simulations [24], [26], [33], [63], [64], and channel traces from real mmWave networks [163]. Other papers propose new CA algorithms, e.g., based on ML approaches that try to predict the behavior of mmWave channels [41], [164]–[166].

The results presented in these studies show that there is no clear “leader” that provides the best performance in all scenarios. There is a tradeoff between throughput and latency.

To demonstrate this tradeoff, we use the NS-3 simulator that contains the models of the most widely used CA algorithms.

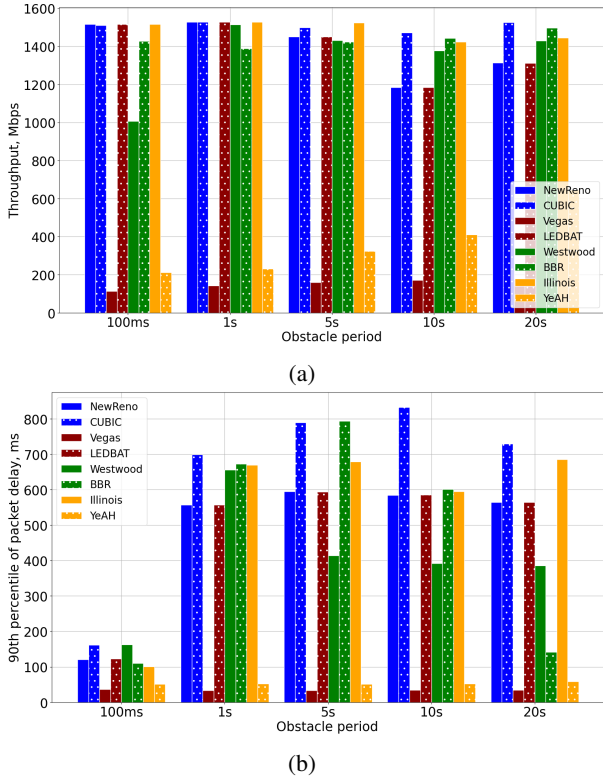


Fig. 9: Evaluation of CA algorithms: (a) average throughput, (b) 90th percentile of latency. RTT between server and gNB is 30 ms.

We run a scenario in which the mmWave link between gNB and UE experiences periodic LOS/NLOS transitions due to periodically appearing obstacles (see the detailed description in Section III-C). In Fig. 9, we plot the average throughput of a bulk TCP flow and the 90th percentile of the latency experienced by packets generated by the ping application. We can see that, for example, the CUBIC algorithm provides high throughput for all the considered scenarios (obstacle periods), but it suffers from the bufferbloat problem. In contrast, Vegas and YeAH achieve low latency, but their throughput significantly degrades. Some algorithms can provide a good balance between throughput and latency for some scenarios (e.g., see BBR performance for low and high obstacle periods), but their performance degrades for other scenarios (see high latency of BBR for 5s period). Thus, it is still an open research question whether it is possible to design a CA algorithm that can achieve Pareto improvement in terms of throughput and latency over highly varying mmWave links.

In the following sections, we show that even with the existing CA algorithms, it is possible to achieve high throughput and address the bufferbloat problem using solutions working at other layers of the protocol stack, e.g., active queue management working at the link layer and cross-layer solutions.

2) *Applicability to QUIC*: As mentioned in Section II, TCP and QUIC have similar congestion control algorithm architecture, thus, a plethora of congestion control algorithms developed for TCP and described in the previous section can

be easily adapted for QUIC. We refer to Section II-B4 for the detailed comparison of TCP and QUIC congestion control.

3) *Impact of various bands on CCA*: High-frequency bands increase throughput by several orders of magnitude (e.g., up to 100 Gbps for THz band [133]). As the end-to-end latency cannot be reduced by the same order, TCP & QUIC will experience much higher BDP that quickly fluctuates with time, especially for THz and LW bands. As the existing Slow Start and CA algorithms are too slow to quickly adapt CWND, many channel resources will be underutilized (see Fig. 4).

Apart from that, long link blockages, which are inherent to THz and LW, may induce multiple RTOs and significantly degrade performance. Note that this problem is less relevant to mmWave bands, in which the NLOS path can still provide considerable throughput.

The mentioned problems can be addressed by: (i) multi-connectivity solutions described in Section VII-A that allow aggregating THz/LW links with low-frequency ones, and (ii) cross-layer interaction between the transport layer and the link layer of high-frequency base stations as discussed in Section VI-C.

B. Active Queue Management

The results presented in Section III-C show that TCP performance significantly depends on how the bottleneck link queue is managed. The simplest queuing algorithm, First-In-First-Out (FIFO) with Tail Drop policy, works as follows. A device keeps a single queue (buffer) having a limited size, either in bytes or in the number of packets. When the device accesses the medium (wired or wireless), it pulls packets from the head of the queue. Packets arriving from the upper layers are put at the end of the queue. The queue length grows if the device pulls packets from the queue at a rate lower than the packet arrival rate. Eventually, no free space is left in the queue, and newly arriving packets are dropped. Currently, vendors prefer to keep large buffers on wireless devices. For example, as demonstrated in Fig 5, this strategy allows fully utilizing the available wireless link capacity. However, when the link capacity suddenly drops (e.g., because of LOS to NLOS transition), packets stay too long in the queue leading to the bufferbloat problem. The researchers proposed various Active Queue Management (AQM) algorithms to address this problem. In contrast to the simple “passive” FIFO Tail Drop algorithm, AQM algorithms can: (i) start dropping packets well before the queue overflows, which is treated by responsive TCP flows as congestion indication, (ii) shape the load of unresponsive UDP flows, (iii) prioritize packets of interactive applications over bulk TCP flows, etc. During the last three decades, many AQM algorithms have been developed. Most of them have been implemented in Linux as part of the so-called Traffic Control Layer (TCL). Below, we shortly describe the TCL architecture and classify various AQM algorithms.

1) Approaches to managing queues:

a) *TCL architecture*: As Fig. 10 shows, TCL operates between the network layer (i.e., the IP protocol) and the link

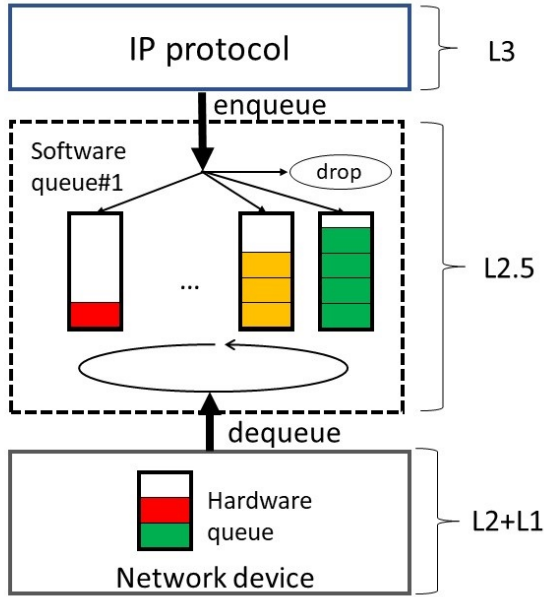


Fig. 10: Traffic control layer architecture.

layer of a network device. So, it is referred to as a Layer 2.5 (L2.5) solution and it works as follows.

Consider a device with RAT-specific links. The device keeps one or several hardware queues⁶ from which it can immediately pull packets when the device accesses the medium. These queues are typically implemented in hardware as FIFO queues. The queue sizes are selected such that the amount of buffered data is enough to use the capacity of the corresponding link when the device asynchronously accesses the medium without inducing additional delays. All other packets that cannot be accommodated in hardware queues are kept at TCL.

TCL can be considered as intermediate packet storage between the network layer and the physical interface. TCL keeps software queues that are managed according to some AQM algorithm (in Linux notation, it is called queuing discipline). When a packet arrives from the network layer, it is put into a specific software queue or dropped depending on the used queuing discipline.

The network device monitors the occupancy of the hardware queue(s). When it detects a free space in a hardware queue, it calls the dequeue function of the corresponding queuing discipline that performs the following actions. First, it selects the software queue(s) from which it pulls packets (i.e., executes the scheduling function). Second, the queuing discipline decides which packets to forward to the network device and which packets to drop (e.g., because of their lifetime expiration).

AQM algorithms proposed so far differ in how many queues they use and how they implement enqueueing and dequeueing functions. Let us consider them in more detail.

⁶For example, Wi-Fi devices keep four queues having different priorities to access the wireless medium.

b) Single-queue algorithms: In this type of algorithms, packets of different flows share a single software queue and are processed according to the FIFO principle. Thus, the AQM algorithm should only decide when and which packets to drop.

One of the oldest AQM algorithms designed to improve TCP performance is Random Early Detection (RED) [167]. As the name states, RED starts to drop packets earlier than the queue becomes full. Specifically, each incoming packet is dropped with some probability that depends on the average queue length. When the queue is almost empty, this probability is equal to zero. Starting from some queue length, it increases linearly. When the queue is full, packets are dropped with a probability equal to one. In contrast to Tail Drop, RED provides earlier congestion signals to the TCP sender that allows reducing the TCP sending rate without inducing massive packet losses. The main problem of RED is that its performance is significantly influenced by the parameters that determine the slope of the linear drop function. Numerous modifications of RED were proposed (see the detailed overview in [88]), among which: (i) Adaptive RED [89] that dynamically selects the parameters of the drop function, (ii) Nonlinear RED [90] in which linear function is replaced with quadratic one. However, these modifications have not become widespread because of their unpredictable performance in terms of queuing delay and unfairness issues between TCP flows that use different CA algorithms [91].

To provide limited queuing delay Controlled Delay Management (CoDel) algorithm was recently developed [92], [93]. For each packet, CoDel estimates its queuing delay based on the timestamp added to the packet by the enqueueing function. CoDel monitors the minimal queuing delay D_{min} observed during time interval T . Initially, T is set to 100 ms. When D_{min} becomes greater than the target queuing delay D_{tar} (by default, $D_{tar} = 5$ ms), CoDel switches to a dropping state. During this state, CoDel drops a packet at the end of each interval $T(N) = T/\sqrt{N}$, where N is the counter increased after each drop. CoDel exits the dropping state when D_{min} goes below D_{tar} , and resets N to one.

The benefit of the described above CoDel dropping rule is that it allows signaling TCP sender(s) to reduce load without losing many packets. However, this rule is too gentle for unresponsive flow, e.g., UDP flows that do not implement any CA algorithm. To solve this problem, the researchers have proposed COBALT (CoDel and BLUE Alternate) that combines CoDel and BLUE algorithms [94]. First, COBALT changes the behavior of CoDel when it exits the dropping state. Instead of resetting N to one, it gradually decreases N after each interval when D_{min} is below the target delay. Second, COBALT uses an additional dropping rule that follows the BLUE algorithm. The BLUE algorithm [95] drops packets with some probability p that depends on the queue length. If the queue is empty for a long time interval, p is set to zero. If during a fixed interval T_{BLUE} (by default, $T_{BLUE} = 100$ ms) packets are dropped because of the queue overflow, p is increased by some fixed increment. Otherwise, if no packets are dropped during interval T_{BLUE} , p is decreased by some

fixed decrement. Authors of COBALT show that it effectively combats the bufferbloat problem in case of unresponsive UDP flows while providing the same performance as CoDel for responsive TCP flows.

The main problem of single-queue AQM algorithms is that all flows, even if they require different service rates, share the same queue. Thus, rarely arriving packets of delay-sensitive flows should wait until packets of other flows (e.g., a bulk TCP flow) are served.

c) Multi-queue algorithms: Multi-queue algorithms use several queues to separate packets of different flows. In addition to the drop function, multi-queue AQM algorithms define specific classification and scheduling functions.

The classification function determines in which queue an incoming packet shall be put. Typically, it is based on the packet headers, e.g., IP and UDP/TCP headers. Several approaches are used to classify packets. The first one is to define several traffic classes, e.g., based on the Type of Service (ToS) field of the IP header. The classification function simply determines the relation of traffic class to a specific queue. The second approach used in modern AQM algorithms, e.g., in a multi-queue variant of CoDel called Flow Queue CoDel (FQ-CoDel) [96], is to allocate a separate queue for each flow. To reduce complexity, this approach is typically implemented as follows. A high number of queues (e.g., 1024) is created. Based on pair of source and destination IP addresses and pairs of TCP/UDP ports, a hash function determines a queue for a packet. Assuming a low number of concurrent flows and low collision probability of the used hash function, each flow occupies a separate queue. The scheduling function determines the order in which packets are pulled from different queues. Existing scheduling algorithms can be split into three groups.

The first one, called Fair Queueing (FQ), aims to provide equal shares of the link capacity to all queues. For example, FQ-CoDel uses the Deficit Round Robin (DRR) scheduler [97], [98], which dequeues the same number of bytes from each active queue within a given time interval. Other examples of FQ-based schedulers are Stochastic Fairness Queueing [99] and Quick Fair Queueing [100].

The second group is called Strict Priority Queueing (SPQ) [101]. With SPQ, all queues are sorted according to their priority that is determined, for example, based on the served traffic class. A non-empty queue with the highest priority is served first. Only when a high-priority queue is exhausted the next queue having a lower priority is served.

The third group, called Weighted Fair Queueing (WFQ) [102], [103] is an extension of FQ that allows various queues to obtain different shares of the link capacity. Specifically, the share of the link capacity obtained by a queue is proportional to the weight assigned to this queue that, in turn, is determined based on the served traffic Quality of Service (QoS) requirements. As intermediate nodes (e.g., base stations, routers) typically have no information about QoS requirements of various flows, FQ is the most widely used scheduler type for serving TCP traffic.

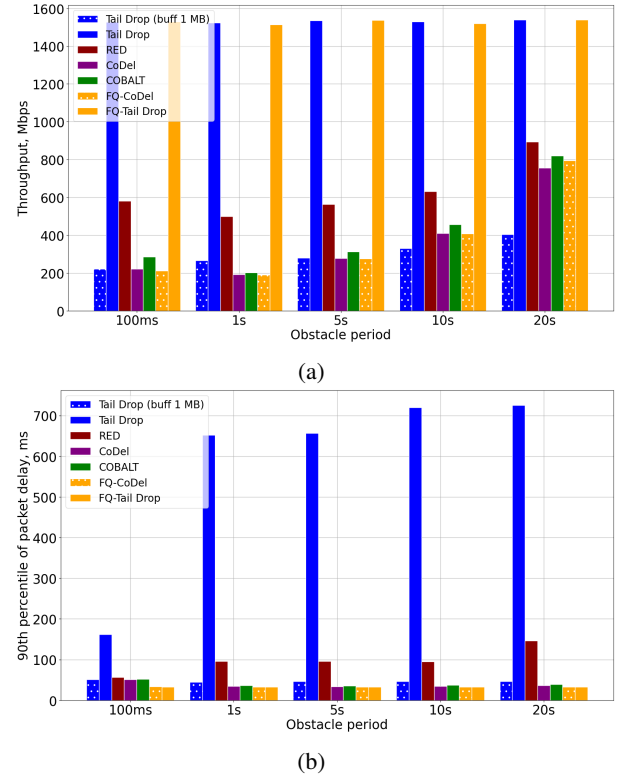


Fig. 11: Evaluation of AQM algorithms: (a) average throughput, (b) 90th percentile of latency. RTT between server and gNB is 30 ms.

The benefit of multi-queue AQM algorithms with the FQ scheduler is that they provide short-term fairness between TCP flows that generate different amounts of traffic. For example, if a bulk TCP flow generated by a video application and an interactive TCP flow generated by a messenger share the same queue, a packet from the interactive TCP flow is put at the end queue. Thus, the packet shall wait until all currently enqueued packets from the bulk TCP flow are served. If different queues are used for the flows, a packet from the interactive TCP flow is served almost instantly. Thus, multi-queue AQM algorithms are promising to avoid the bufferbloat problem for a mixture of bulk and interactive TCP flows.

2) Evaluation of AQM algorithms: In Fig. 11, we compare the performance of various AQM algorithms. Similar to Section VI-A, we consider a scenario in which a mmWave link between a gNB and a UE experiences periodic LOS/NLOS transitions. Two flows are transmitted in downlink: (i) a bulk TCP flow for which we measure the average throughput, and (ii) an interactive TCP flow periodically generating small packets for which we measure one-way transmission delay. Unless otherwise explicitly stated, the size of the buffer at gNB is set to 30 MB, which is higher than the maximum BDP. The TCP sender uses the CUBIC CA algorithm.

Let us first consider the results for the single-queue AQM algorithms that put packets of different flows in the same queue. Depending on the buffer size, the Tail Drop policy either maximizes throughput but suffers from the bufferbloat

problem or provides low latency and low throughput (see the results for 1 MB buffer). To keep queuing delay low, other algorithms (RED, CoDel, COBALT) drop several packets when the link transits to the NLOS state, significantly reducing CWND for a bulk TCP flow. However, when the link returns to the LOS state, the amount of buffered data is not enough to fully use the available link capacity. Similar observations for single-queue AQM algorithms were reported in other studies (e.g., see [26], [63], [168], [169]).

In contrast to these studies, we also evaluate multi-queue AQM algorithms: (i) FQ-CoDel, which keeps separate queues for different flows and applies CoDel packet dropping policy for each queue, (ii) FQ-Tail Drop, which is a modification of FQ-CoDel with D_{tar} set to infinity (i.e., packets are dropped only when the total amount of enqueued data exceeds buffer size). We can see that the usage of separate queues provides minimal delay for the interactive flow. As for bulk TCP flow, its throughput is similar to that of the corresponding single-queue algorithm. Thus, proper configuration of multi-queue AQM algorithm at the bottleneck link can provide Pareto improvement in terms of throughput and latency compared to the cases of single-queue AQM algorithms and various CA algorithms that can be used at TCP sender (see also the results in Fig 11). It should be noted that in cellular systems, the gNB already keeps separate queues for each UE and use specific schedulers to share wireless channel resources between UEs. However, TCP flows belonging to the same UE are typically put in a single FIFO queue leading to the intra-UE bufferbloat problem. As shown above, a promising solution is to use the FQ-Tail Drop queuing discipline that provides short-term fairness between different flows of the same UE.

3) *Applicability to QUIC*: Since AQM is implemented at the link layer, there is no direct dependency between the used algorithm and the transport layer protocol. In other words, the same algorithm can be applied for both TCP flows and QUIC flows that are encapsulated in UDP flows. RFC 7567 [170] providing IETF recommendations regarding AQM divides data flows into two categories: (i) TCP-friendly flows that reduce their sending rate in response to congestion notifications (e.g., packet drops), (ii) non-responsive flows that do not adjust its rate in response to congestion notification. QUIC flows correspond to the first category. Thus, the AQM algorithms designed for TCP can be applied to QUIC. Moreover, a QUIC flow can be identified by the destination port field in the UDP header. Specifically, port 443 is assigned for QUIC.

4) *Impact of high-frequency bands on AQM*: As high-frequency bands increase BDP, the AQM approaches require larger buffers (e.g., an order of GB for THz/LW bands), which increases the complexity of base stations. However, the small coverage, especially for THz/LW bands, and a high density of base stations limit their cost and complexity. One of the approaches to solve this problem proposed in [171] is called Virtual AQM. With this approach, a part of the buffer can be kept at other devices located close to base stations. However, the open question is how to manage buffers distributed over several nodes. Another approach is to use cross-layer solutions

that adapt CWND based on the current estimation of BDP. If CWND is selected close to BDP, the traffic and link quality fluctuations can be smoothed with the small buffer.

C. Cross-layer solutions

As mentioned in Section II-A, because of transport layer abstraction, the TCP sender considers the whole path between endpoints as a black box and does not know the characteristics of each intermediate link. Thus, the TCP sender can use only local measurements to select transmission parameters (e.g., CWND and SSThr) and employ sophisticated congestion control algorithms considered in Section VI-A to adapt these parameters to varying characteristics of specific intermediate links (e.g., mmWave links).

One of the promising ways to improve TCP performance widely studied in recent years is to use cross-layer interaction. The idea is to obtain additional measurements at intermediate network nodes (e.g., router, gNB) operating at the physical/link layer of the protocol stack and provide them to the TCP sender operating at the transport layer. The main problem is how to organize such cross-layer interaction between different layers of the protocol stack, especially when these layers are operating at different physical devices. Below, we consider three approaches to provide cross-layer interaction: (i) in-device interaction, (ii) interaction via TCP/IP header modifications, and (iii) usage of special service protocols.

1) Cross-Layer Approaches:

a) *In-device interaction*: With in-device cross-layer interaction, the TCP sender is located on a device (e.g., UE or gNB) that can directly measure the characteristics of the wireless link. In particular, several works [104]–[106] consider uplink data transmission when a UE is the TCP sender. In such a case, it is possible to deliver physical and link-layer measurements (e.g., link capacity) to the transport layer. Assuming that the wireless link is a bottleneck, the TCP sender can set CWND to the current BDP estimation and, thus, fully use the available link capacity.

In the case of a downlink transmission, the authors of [79] propose to use a split PEP approach according to which the TCP sender of the second TCP connection (see Fig. 8(a)) is located at the gNB or close to the gNB. Similar to the uplink case, it can use channel measurements available at the gNB to estimate BDP and properly select CWND.

b) *Interaction via TCP/IP header modification*: Another way to implement cross-layer interaction is to use TCP/IP headers. With this approach, the control information, such as congestion notification or link capacity measurements, can be delivered to a TCP sender using special fields in the TCP/IP headers of the corresponding data and ACK packets. The intermediate network devices can modify these fields to include the actual control information.

A simple example of such cross-layer interaction is the Explicit Congestion Notification (ECN) mechanism specified in [107]. If endpoint devices and an intermediate device support the ECN mechanism, the intermediate device can signal congestion events to the TCP sender by setting appropriate bits

in the TCP/IP headers without packet dropping. The support of the ECN mechanism by all intermediate devices allows avoiding packet losses and performance degradation caused by excessive retransmissions of lost packets during the execution of Fast Retransmit and Fast Recovery algorithms [172].

Another existing example of “in-flow” cross-layer interaction is the Quick-Start mechanism specified in [108]. Quick-Start uses the TCP options that are special containers piggybacked to the TCP header. The idea of the Quick-Start mechanism is to determine the bottleneck link capacity during the TCP connection establishment phase. Specifically, in the first packet of a TCP connection, the TCP sender adds the Quick-Start Request option that contains the value of the expected data sending rate (typically, it is set to a sufficiently high value). The intermediate device that receives a packet with the Quick-Start Request option checks whether it can support the requested sending rate. If the corresponding link data rate is lower than requested, it can reduce the value indicated in the Quick-Start Request. Thus, hop-by-hop traversing the path to the TCP receiver Quick-Start Request option will contain the value of the bottleneck link data rate. The TCP receiver sends back the obtained minimal value of the sending rate in the Quick-Start Response option. As the result of the described procedure, the TCP sender obtains the data rate of the bottleneck link, and the estimation of RTT needed to calculate BDP. Further, the obtained value of BDP can be used as the initial value of CWND and SSThr.

The authors of Quick-Start note that usage of a high initial value of CWND can cause the transmission of a burst of packets that may lead to massive packet losses. To avoid this problem, they recommend using the TCP pacing feature [173] according to which packets are sent with some delay. Specifically, the delay between two consequent packets can be selected as the ratio of RTT and the selected packet sending rate. Authors of Quick-Start show that implementation of this mechanism can address the Slow Start problem and increase resource utilization in networks with high BDP-links [174].

Although the described above mechanisms have clear benefits, they have not obtained wide deployment for the following reasons. First, to make these mechanisms work, all or almost all intermediate devices on the Internet shall support novel mechanisms, which is impossible due to high CAPEX/OPEX and wide heterogeneity of Internet devices, their vendors, and owners. Second, as detailed in [35], some intermediate devices called middleboxes (e.g., old firewalls) can modify TCP/IP headers by cutting off unrecognized/unsupported options or even can drop such packets, thus, vanishing all the benefits from the novel mechanisms.

c) Interaction via special service protocols: One of the approaches to avoiding the harmful influence of middleboxes is to use special service protocols. These protocols create a tunnel (e.g., a TCP connection) between two devices that can be used to transmit cross-layer information. An example of such a cross-layer protocol called xStream is proposed in [109]. xStream enables the communication between a 5G system and endpoint devices (e.g., a UE and a server). Specif-

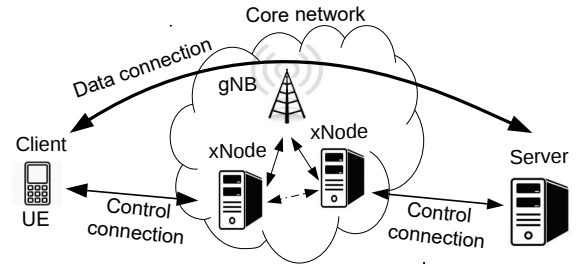


Fig. 12: xStream architecture [109]

ically, an operator of a 5G system deploys in its core network a special logical node called xNode (see Fig. 12). With a special DNS query, endpoint devices can obtain the network address of xNode and establish a TCP connection with xNode. This TCP connection can be used to exchange various types of control information between the 5G system and endpoint devices. For example, xNode can gather information about the channel capacity of the gNB serving a particular pair of endpoints and send this information to them. Thus, similar to Quick-Start, a TCP sender located at the server can use this information to estimate BDP and properly select CWND and TCP pacing rate. A similar approach is considered in [110]. However, the authors do not describe the details of the used service protocol.

The benefit of the described above approach based on novel service protocols, such as xStream, is that it does not require all the devices on the data path to support this novel protocol. Thus, this approach has significantly lower implementation complexity than the approach introducing new TCP options. Moreover, information transmitted over a separate control TCP connection is protected from middlebox influence. However, the information provided by xNode can be limited to specific links (e.g., wireless links) that are assumed to be bottlenecks. Thus, the TCP sender shall still use legacy congestion control algorithms. For example, it shall react to packet losses that may occur outside the 5G system. So, the obtained control information should be considered as a recommendation to assist CWND choice rather than direct command.

2) Applicability to QUIC: As QUIC does not allow intermediate nodes to modify its headers, the solutions described in Section VI-C1b based on the cross-layer information provided in the TCP option field cannot be adapted for QUIC. However, the ECN mechanism that uses the ECN flag in the IP header is supported by QUIC [16]. The in-device interaction and interaction between wireless devices and QUIC sender/receiver requires the design of special cross-layer protocols. An example of such a protocol can be found in [28].

3) Benefits and drawbacks of cross-layer solutions with various bands: Let us consider how cross-layer solutions can improve TCP & QUIC performance. Figure 13 shows the results of an experiment with a single mmWave gNB serving several UEs downloading files. The file size is a random variable drawn from a truncated lognormal distribution with the average value of 10 MB, minimal and maximum values

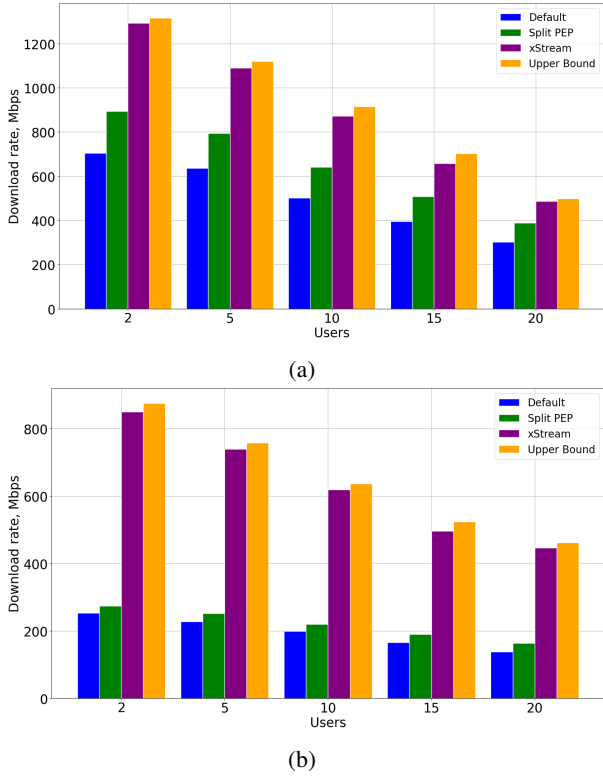


Fig. 13: Evaluation of solutions for the Slow Start problem: (a) RTT=5 ms (b) RTT=30 ms.

of 1 MB and 100 MB, respectively. We plot the average file download rate, defined as the ratio of the file size and the total time required to download the file. We also consider an “Upper bound” case in which the server uses infinite initial CWND, and all intermediate devices have infinite buffers. In this case, after a connection setup, a file is immediately put into the gNB buffer without a long Slow Start procedure. We can see that the cross-layer information provided by the xStream protocol significantly increases file download rate close to the upper bound. The gain becomes even higher for higher RTT between the server and the gNB. For comparison, we also plot the results for the split PEP solution. We can see that split PEP increases the download rate when the RTT between the server and the gNB is low, and it does not change the download rate when the RTT is high. So, we can conclude that the implementation of novel service protocols that provide additional cross-layer information is a promising way to address the Slow Start problem and improve TCP & QUIC performance over high-frequency wireless links.

The main drawback of cross-layer solutions is their complexity. Switching from mmWave to THz/LW bands significantly increases the throughput, which means that the base station requires to process more and more packets, e.g., parse and modify their TCP/IP headers.

Many cross-layer solutions are based on the estimation of the wireless link capacity which is then provided to the transport-layer protocols to select transmission parameters (e.g., CWND). Since this information is delivered with some

delay it may become outdated. This problem is especially important for THz/LW bands in which throughput can quickly fluctuate by several orders of magnitude. Thus, an important problem for future research is how to design effective transmission parameters selection algorithms in case the input data (e.g., the link throughput) significantly changes with time and is hardly predictable.

VII. MULTIPATH COMMUNICATIONS

Multipath communications is a promising approach to improve TCP/QUIC performance in high-wireless systems because the usage of several physical paths between a server and a client may increase both throughput and reliability. A key problem of multipath communications is how to control the transmission of data over several network paths. This problem can be solved: (i) at the data link and physical layers of wireless devices with a special feature called Multi-Connectivity and considered in Section VII-A, (ii) at the transport layer with the multipath extensions of TCP/QUIC that are analyzed in Section VII-B.

A. Multi-connectivity

Multi-Connectivity (MC) is an essential feature of modern wireless networks that allows a client device to connect to several serving base stations simultaneously. It is supported by Wi-Fi, various 3GPP technologies, and even a mixture of 3GPP and non-3GPP ones. MC is beneficial for mmWave, THz, and Light communications, as it: (i) increases the overall capacity between endpoints and (ii) allows overcoming outages of high-frequency links. This section considers how MC is implemented in various wireless technologies and estimates the time such implementations require to react to quality fluctuations of high-frequency wireless links.

MC can be implemented in wireless networks at different levels of integration between the links, which affects the flexibility of the resource utilization, packet losses in case of link blockage, achieved throughput, and delays required to switch between various links.

One option is that wireless technologies provide several service access points (SAP) to the upper-layer protocols, each corresponding to a specific wireless link. Although such an approach is easy-to-implement, it requires transport-layer protocols or applications to manage data transmission via various links. Thus, such protocols shall be capable of balancing the load over multiple connections and processing the events when some packets are lost or delayed at a link because of its blockage or congestion. As transport-layer protocols do not directly interact with wireless links, they slowly react to the changes in wireless link properties, which degrades performance, as discussed in detail in Section VII-B.

During the past decade, many efforts have been made towards another option, native support of the multi-connectivity transparent for the upper-layer protocols.

Below, we review various solutions and evaluate their impact on TCP & QUIC and overall system performance, see Table III.

TABLE III: Summary of Multi-Connectivity solutions

Layer	Solution	Packet retransmission delay	Complexity	Spectrum Efficiency	Initially proposed
System/Network	ANDSF	seconds	Low	Low	3GPP Rel. 8
PHY	CoMP, later: Distributed MIMO	instant (no retransmission)	High	High	3GPP Rel. 11, a candidate feature for Wi-Fi 7/8
PHY and MAC	Relaying	< 5 ms	Medium	Low	Link Switching and link cooperation in 802.11ad
MAC	Carrier Aggregation	< 10 ms	High	High	3GPP Rel. 10
MAC	FST	< 10 ms	Medium	Medium	mmWave Wi-Fi
MAC	Multi-Link	< 5 ms	High	High	Wi-Fi 7
PDCCP	Dual/Multi Connectivity	10...100 ms	Medium	Medium	3GPP Rel. 12

1) Approaches to Multi-Connectivity:

a) Access Network Discovery and Selection Function:

One of the first 3GPP solutions toward multi-connectivity is Access Network Discovery and Selection Function (ANDSF), an entity in the core network supporting intelligent offload between 3G/4G to Wi-Fi and vice versa [175]. Notably, this solution is designed to manage long-term paths; it cannot save the packets on the fly if a link is blocked. So it is inefficient in case of frequent switching between the links.

b) *Coordinated Multi-Point*: To overcome link blockage and poor channel conditions, Coordinated Multi-Point (CoMP) — or Distributed Multiple-Input Multiple-Output (MIMO), to which it has evolved — allows multiple base stations to simultaneously transmit the same data to a UE with poor channel conditions (or to receive data from it) to improve transmission reliability. CoMP is implemented in LTE Rel. 11 and was considered a candidate feature of Wi-Fi 7, which is currently under development. Although initially proposed for low-frequency communication, CoMP is recently shown to be fruitful in solving the link blockage problem in mmWave 5G systems [176]. The probability of link outage is significantly reduced thanks to multiple nodes simultaneously transmitting or receiving the same data. Even if a link to one base station is lost, the other ones will deliver the data without additional delays and, thus, without any impact on the transport-layer protocols. The cost for such reliable delivery is enormous overhead and the necessity of extremely wide backhaul connections between base stations. It is the main reason why CoMP is unlikely to be supported by upcoming Wi-Fi 7 technologies despite being intensively discussed.

c) *Relays*: The mentioned above overhead can be reduced by the usage of relays that are involved in the data delivery process only when the main link degrades. For example, in mmWave Wi-Fi networks (IEEE 802.11ad), relaying allows a source STA S to transmit packets to a destination STA D with the assistance of another STA R called relay in case of blocked link $S \rightarrow D$. IEEE 802.11ad defines two types of relay operation.

Link switching. If the link between S and D is blocked, S can redirect packets destined for D to relay R , which, in turn, forwards packets to D . Depending on the capabilities of R , particularly the number of antennas and RF chains, R may operate either in the Full-Duplex Amplify-and-Forward (FD-AF) mode or in the Half-Duplex Decode-and-Forward (HD-DF) mode that increases transmission duration.

Link cooperation. In this case, R is actively involved in

direct communication between S and D . For that, first, S sends the data packet to R . Then S and R simultaneously transmit the copy of this packet to D , which increases the received signal strength at D and therefore increases the probability of successful delivery of the packet.

In both cases, LOS path blockage of a mmWave link reduces the link capacity and increases transmission duration but does not cause packet loss.

d) *Carrier aggregation*: In cellular systems, the tightest and the most efficient cooperation between several wireless links established between the same pair of devices (the base station and the user equipment) is achieved with carrier aggregation, Introduced in Rel.10 and operating at the Medium Access Control (MAC) layer, Carrier Aggregation allows a UE to be simultaneously connected with multiple links at different frequencies to the same base station. Thus, in theory, the base station can perform cross-link scheduling which simplifies load balancing between the links and allows reacting to channel quality degradation as soon as the MAC layer acknowledgment is received, i.e., with a delay of several ms. In reality, the usage of Carrier Aggregation in 5G systems is complicated for the following reasons⁷. First, various 5G bands may use different channel numerology, i.e., in various bands, the time slot may equal 1 ms, 0.5 ms, 0.25 ms, etc. Second, low-frequency and high-frequency bands may have different coverage and different channel quality, which may limit the usage of some bands by cell-edge users. Third, using multiple links increases the power consumption of mobile devices. Fourth, the total amount of resources to be scheduled is so large that it is worth performing scheduling separately in each band to reduce computational complexity. Fifth, if a packet sent over one link is not delivered, the following data delivered on the other link are buffered at the receiver without forwarding to the upper-layer protocols.

e) *Multi-link*: A technique similar to Carrier Aggregation, called Multi-Link, is being developed for Wi-Fi 7 networks⁸. Although the standard developers assume that packets sent via one link can be acknowledged and retransmitted via another one, in reality, such an opportunity may be limited. Although the standard claims that the transmission occurs several

⁷As far as we know, by now, these issues block the aggregation of both low-frequency and high-frequency bands. In December 2020, Ericsson announced the first cross-band aggregation of low-frequency and mid-frequency bands (410 MHz–7125 GHz) [177]. At the same time, carrier aggregation for either low-frequency or high-frequency bands is already supported.

⁸Right now only low-frequency bands are considered.

dozens of microseconds after the channel becomes idle, Wi-Fi devices shall prepare an aggregated packet, perform channel measurements, etc., well before the actual transmission. These peculiarities may limit the ability to instantly retransmit a non-delivered packet over another link, though no losses are expected, and the induced delays may be less than 5ms, which has a negligible impact on TCP. That is why the developers of the IEEE 802.11bb standard for Light Communications predict that future versions of the standard will allow the usage of Multi-Link to improve light communications.

It is expected that Multi-Link will replace the Fast Session Transfer (FST) mechanism that was designed for IEEE 802.11ad to allow seamless handover between mmWave and low-frequency links such that transfer time does not exceed 5 to 10 ms. FST can be implemented in two ways. The first one assumes a single MAC interface. So, FST runs seamlessly for the high-level protocols. The second one provides several interfaces to the upper layers, and the higher layers are responsible for managing the session transition between different frequency bands.

f) Dual Connectivity: As for cellular systems, the most flexible and powerful existing solution is Dual Connectivity (or Multi-Radio Dual Connectivity). This feature has appeared in LTE Rel.12, works at the Packet Data Convergence Protocol (PDCP), and enables aggregation of two radio links between a UE and two base stations (which differs from Carrier Aggregation, where there is a single base station). The latter aspect is very important for aggregating several bands with different coverage, as the corresponding base stations can be deployed separately with different densities. Thus, high-frequency base stations may be closer to the users. DC supports different radio access technologies. Since the traffic is split between the links at the PDCP layer, lost packets may be retransmitted, but there is no easy way to move buffered packets from one link to another, as it is done with Carrier Aggregation if a link quality degrades. Consequently, the following approaches have been proposed to split the traffic.

Packet Duplication. With this approach, the same packets are sent via several links, which increases reliability, reduces latency by eliminating packet retransmissions, improves mobility robustness, and reduces losses of packets as they are stored at several base stations [178]–[180]. However, it multiplies radio resource consumption and thus reduces the throughput [181], [182].

Packet Splitting. With this approach, packets are sent over the different paths, e.g., proportionally to the capacities of the links [183], though other approaches can be used. Packet Splitting increases throughput as the UE receives packets from multiple nodes and reduces delays in comparison to the case when all the packets are sent on the fastest link. At the same time, the reliability is not increased, and high-frequency link blockage may lead to packet loss that affects the TCP performance [184].

Network Coding (NC). With this approach, different encoded combinations of packets are sent on multiple paths. For example, Random Linear Network Coding (RLNC) is often

used because it provides a good trade-off between bandwidth efficiency, complexity, and delay, compared to other network coding or forward error correction strategies [185], [186]. Network coding may balance the advantages of the other approaches. However, it requires more sophisticated multi-connectivity control algorithms, e.g., the paper [187] proposes a greedy heuristic to control transmitting data with dual connectivity and NC, which minimizes delivery delays.

2) Applicability to QUIC: All MC solutions described above are applicable for both TCP and QUIC. However, QUIC will provide better performance with ANDSF that splits the data at the network layer. With ANDSF, wireless interfaces of a UE have different IP addresses. Thus, a TCP/QUIC data flow can be served with one interface at a time (this problem is addressed by multipath versions of the corresponding protocols, see next section). When a TCP/QUIC flow is redirected from one wireless interface to another, TCP shall reestablish the connection because the UE changes its address. In contrast, QUIC does not need to reestablish the connection thanks to the “0-RTT connection setup” feature as described in Section II-B. Other MC solutions working at the link and the physical layers are transparent for TCP/QUIC flow.

3) Impact of various bands on Multi-connectivity: Advanced physical layer (PHY) MC solutions (e.g., CoMP) require high computational power at base stations and the so-called ideal backhaul (i.e., very low latency and high-throughput links between base stations). While such solutions are currently considered for mmWave bands [176], their usage in THz/LW bands is questionable because of the much higher density of THz and LW base stations and their expected low cost and complexity.

One of the promising MC solutions for high-frequency bands is to keep several links with the base stations operating in various bands (e.g., in LW and low-frequency bands). When the LW link is blocked, traffic is quickly forwarded to the low-frequency link. Thus, an extension of 3GPP Multi-radio Dual Connectivity and Wi-Fi Multi-link solutions to THz/LW bands can efficiently address the link blockage problem.

B. Multipath TCP/QUIC

1) General Idea: Several multi-connectivity solutions described in Section VII-A require that upper-layer protocols manage the usage of various links. It can be done with Multipath TCP (MPTCP) [188], [189] or Multipath QUIC (MPQUIC) [71], [190], which have been developed to control data transmission if several paths exist between the two endpoints. Both MPTCP and MPQUIC identify each path by a pair of sockets that combines IP addresses and ports. Thus, they work only if at least one device, e.g., the client, knows that there are several paths between the communicating nodes. For example, a UE may have two radio interfaces for high-frequency and low-frequency bands with different IP addresses. Also, both MPTCP and MPQUIC allow a server to advertise additional addresses and ports on which it can be reached [188], [190].

MPTCP and MPQUIC are designed to improve throughput compared with single-path transport protocols that use the best available path and provide dynamic load balancing by selecting a less congested path while not harming legacy flows.

2) *MPTCP*: MPTCP inherits the functionality of traditional TCP and is backward compatible in the following sense. MPTCP provides a single interface to the application. So, for the application, MPTCP operates in the same way as usual TCP. Each MPTCP connection consists of one or several single-path TCP connections referred to as subflows.

To achieve data integrity on one side and to follow legacy per-subflow congestion control, each portion of data is assigned two sequence numbers. The first one, called multipath sequence number, is used for segmentation and reassembly of the data as well as retransmissions via another subflow. The second one, the traditional TCP sequence number, is used for congestion control. The multipath sequence number is transmitted as a TCP option, while the subflow ones are transmitted in the original field of the TCP header.

MPTCP uses both MPTCP connection-level and subflow-level acknowledgments to provide a robust data delivery to the application. Thanks to two sequence numbers, MPTCP allows a data segment to be retransmitted on a different subflow from that on which it was originally sent, which is fruitful in the case of high-frequency link blockage. Also, it is suggested to reschedule data sent via one subflow for transmission on different subflows to reduce the delays and avoid connection underutilization when a path breaks. Note that in all cases with retransmissions on different subflows, the lost segments should also be sent on the original path to maintain subflow integrity and backward compatibility with the single-path TCP for networking middle-boxes because they may assume single-path TCP behavior.

3) *MPQUIC*: Introducing multipath operation to QUIC is much easier than to TCP because QUIC does not use IP addresses and ports to identify a flow. Instead, it uses an explicit Connection ID. Even single-path QUIC copes with NAT re-binding, offloading, and IP address updates using its migration feature, but only one path can be used simultaneously. An MPQUIC connection is identified by a set of Connection IDs. As QUIC packets are encapsulated in UDP, which does not require any feedback, MPQUIC allows separating direct and reverse data transmission and using different paths for them. For example, the downlink transmission may use a mmWave link, while the uplink one uses a low-frequency link.

The other basic ideas of operation of MPTCP and MPQUIC are very similar.

4) *Issues induced by MPTCP and MPQUIC and impact of various bands*: In both MPTCP and MPQUIC, congestion control is an important problem. Note that the issue of high BDP and high probability of RTO for THz/LW bands described in Paragraph VI-A3 are also relevant to MPTCP/MPQUIC congestion control algorithms.

As stated in the specifications [188], [190], all subflows belonging to the same MPTCP connection shall be considered one connection. Thus, to be fair with legacy TCP, the CWNs

of the subflows of the same MPTCP connection shall not grow faster than that of the legacy TCP connection.

RFC 6356 [113] proposes a coupled congestion control algorithm, which is shown to be unfair with respect to legacy TCP [114], [115]. The papers [114], [115] propose two algorithms, namely, Opportunistic Linked Increases Algorithm (OLIA) and the Balanced Linked Adaptation Algorithm (BALIA), which address this problem. However, both algorithms are based on the legacy design of Reno and New Reno congestion control algorithms that show low performance with highly dynamic mmWave links [117]. In addition to the aforementioned congestion control algorithms, the current Linux implementation of MPTCP supports mVegas [116]. However, the single-path version of this algorithm shows worse performance on mmWave links [117].

In regular, single-path TCP, it is usually recommended to set the receive buffer twice as large as the Bandwidth-Delay Product. For MPTCP, a subflow packet loss or subflow failure should not affect the throughput of other working subflows. So for MPTCP, the BDP shall be computed using the largest bandwidth and the largest delay among the links. It brings a problem for the joint usage of high-rate mmWave, THz, or LW links together with congested low-frequency links that experience high delays. For example, even for the throughput of 1 Gbps and delays of 40 ms, the buffer shall exceed 10 MB for each MPTCP session, which may be an issue for servers with thousands of connections. Note that in THz bands the throughput is expected to be 100...1000 times higher.

Another issue relevant to MPTCP/MPQUIC is path management (PM), i.e., management of the paths that shall be open between endpoints. Specifically, PM answers the question of how many and which subflows shall be established between two endpoints. For example, if endpoints have two and three interfaces, correspondingly, with a full-mesh strategy, MPTCP/MPQUIC establishes all $2 \times 3 = 6$ subflows which may consume many resources (e.g., memory).

One of the most crucial challenges is the multipath scheduler that distributes data over various subflows. For example, a simple round-robin scheduler cyclically sends packets over different subflows, as long as their CWNs allow sending more data. However, this scheduler does not consider the heterogeneity of high-frequency and low-frequency bands, as well as the delays and losses in each subflow, which significantly degrades performance [191], [192]. To improve performance, various approaches have been proposed in the literature: minimizing delays [193]–[198], blocking avoidance [194], and machine learning approaches [199], [200].

The scheduling problem becomes more complicated if one of the paths traverses via THz or LW link. With a high probability, a THz/LW link may be blocked for a relatively long time interval. The MPTCP/MPQUIC sender should: (i) determine that a path is blocked, (ii) retransmit packets that are assumed to be blocked over another path. The speed of such a reaction is limited by the whole RTT between the user and the server. Note that the usage of MC solutions described in Section VII that split packets at base stations

considerably reduces the control loop and, thus, can much faster forward packets from the high-frequency to the low-frequency link. Therefore, we believe that link-layer MC solutions are more promising for THz and LW bands than the usage of MPTCP/MPQUIC.

VIII. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

In this survey, we analyzed how the most popular transport layer protocols TCP and QUIC would work in future high-frequency wireless systems operating in mmWave, THz, and LW bands. We showed that the common peculiarities of high-frequency wireless links lead to severe performance degradation of TCP & QUIC, i.e., low throughput and high latency.

We thoroughly reviewed various existing solutions aimed at improving TCP & QUIC performance developed by IETF, 3GPP, IEEE, and the research community. Based on the used approach (i.e., network architecture changes, traffic control, multipath communications) and the layer of the OSI protocol stack at which the considered solution is implemented, we divided all found solutions into seven groups. Below, we summarize the benefits and drawbacks of various groups of solutions, their implementation complexity, and possible directions for future research. Also, the key findings of our analysis are outlined in Table II.

The first group of solutions aims at reducing the overall RTT between a user and a server by (i) deploying additional real/virtual servers close to base stations, and (ii) using modern RATs which reduce RTT over a wireless link. Low RTT speeds up the convergence of congestion control algorithms and, thus, efficiently solves the slow start and channel fluctuation problems. However, the implementation of such solutions can significantly increase the operator's CAPEX/OPEX. Further reduction of RTT can be achieved with new RATs (e.g., sixth generation (6G) THz systems and future versions of LW Wi-Fi), which target sub-millisecond latency.

The second group of solutions is based on deploying transport-layer PEPs in the core network, i.e., special nodes that modify TCP headers or shape TCP flows to meet the current wireless link capacity. PEPs can efficiently address the slow start and channel fluctuation problems. However, deploying PEPs also increases CAPEX/OPEX. Also, some PEPs, e.g., those that estimate the capacity of the wireless link, require tight interaction with the serving base station. Currently, many PEPs widely used in practice are designed for TCP. Future research can be connected with designing PEPs for QUIC taking into account QUIC header encryption or extending QUIC to support interaction with PEP.

The third group of solutions uses novel congestion control algorithms at endpoints that aim at increasing throughput and reducing latency. However, recent studies of modern algorithms over high-frequency links show that no single algorithm provides the best performance in all scenarios. A promising direction for future research is to use machine learning for selecting CWND taking into account previous measurements or even predicting the throughput of the high-frequency link. Note that the deployment of new congestion control algorithms

for TCP is a long process because it requires upgrading OSs at servers and users, while the deployment of new algorithms for QUIC only requires application updates.

The fourth group of solutions uses modern AQM algorithms implemented at the link layer of base stations and mobile devices. In particular, as shown in this survey properly configured multi-queue AQM algorithms provide high throughput for bulk TCP & QUIC flows and low latency for delay-sensitive flows irrespective of the congestion control algorithms used at endpoints. Note that for very high data rates (> 100 Gbps typical for THz and LW bands), base stations require buffers of several GB, which increases their complexity and cost. A promising approach recently proposed in [171] is called Virtual AQM. With this approach, a part of the buffer can be kept at other devices located close to base stations. The open research question is how to manage buffers distributed over several nodes.

The fifth group of solutions provides a cross-layer interaction between the link layer of wireless devices and the transport layer of endpoints. In particular, the information about the wireless link capacity provided by a base station to a server can assist CWND selection and, thus, significantly increase throughput and reduce latency. However, the current OSI protocol stack provides very limited capabilities for exchanging information between different layers of various devices (e.g., via headers). Thus, a promising direction is to redesign the protocol stack to enable tight cross-layer interaction between various layers and devices. While currently each layer and a device perform a local performance optimization (e.g., congestion control algorithm selects CWND based on the local measurements), the cross-layer interaction opens the door for global network performance optimization.

The sixth group of solutions uses various multi-connectivity features of the modern RATs. With these features, mobile devices can operate in both high-frequency and low-frequency bands transparently for TCP & QUIC endpoints. If the throughput of the high-frequency link degrades, the whole or a part of TCP & QUIC flow can be quickly switched to the low-frequency link, which efficiently solves the channel fluctuation and link blockage problems. Note that the multi-connectivity feature is implemented independently in each RAT. For example, if a user connects to a 5G base station over a low-frequency link and a Wi-Fi access point over a high-frequency link, the endpoints will see two independent data paths. Therefore, a promising direction for future work is to enable multi-RAT multi-connectivity that will aggregate the wireless links belonging to various RATs and will be transparent for the transport layer protocols.

Currently, the multi-RAT multi-connectivity is managed by the seventh group of solutions, i.e., multipath extensions of TCP & QUIC. However, re-routing the traffic with MPTCP/MPQUIC is very slow because of the high RTT between a user and a server, which increases the retransmission delay. However, the mmWave, THz, and LW devices are typically located close to each other, which provides lower RTT of wireless links (see Table III) than the overall RTT.

Summing up, the results of our analysis show that being deployed and properly configured, existing solutions (e.g., CDNs, PEPs, AQM, multi-connectivity) can substantially improve TCP & QUIC performance in future high-frequency wireless systems albeit various solutions impose different implementation complexity and cost. We identified several directions for future research that are expected to further improve the performance. We believe that enabling tight cross-layer interaction between different layers of the OSI stack and different devices will be an important tool for achieving synergistic effects from the joint usage of various types of solutions and providing global network performance optimization.

REFERENCES

- [1] "Cisco Annual Internet Report (2018–2023) White Paper." [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>
- [2] IEEE, "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Enhanced Throughput for Operation in License-exempt Bands above 45 GHz," *IEEE 802.11ay-2021*, 2021.
- [3] "TS 38.300: NR; NR and NG-RAN Overall description; Stage-2," 3GPP, Dec. 2021.
- [4] IEEE, "Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment: Light Communications," *IEEE 802.11bb*, 2022.
- [5] E. Khorov and I. Levitsky, "Current Status and Challenges of Li-Fi: IEEE 802.11bb," *IEEE Commun. Stand. Mag.*, vol. 6, no. 2, pp. 35–41, 2022.
- [6] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, 2020.
- [7] M. Giordani *et al.*, "Toward 6G Networks: Use Cases and Technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, 2020.
- [8] V. Petrov, T. Kurner, and I. Hosako, "IEEE 802.15.3d: First Standardization Efforts for Sub-Terahertz Band Communications toward 6G," *IEEE Commun. Mag.*, vol. 58, no. 11, pp. 28–33, 2020.
- [9] I. Rodriguez *et al.*, "Radio Propagation into Modern Buildings: Attenuation Measurements in the Range from 800 MHz to 18 GHz," in *Proc. IEEE VTC*, 2014, pp. 1–5.
- [10] H. Zhao *et al.*, "28 GHz millimeter wave cellular communication measurements for reflection and penetration loss in and around buildings in New York city," in *Proc. IEEE ICC*, 2013, pp. 5163–5167.
- [11] M. Polese *et al.*, "Improved Handover Through Dual Connectivity in 5G mmWave Mobile Networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 9, pp. 2069–2084, 2017.
- [12] M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Mobility Management for TCP in mmWave Networks," *Proc. mmNets*, pp. 11–16, Oct. 2017.
- [13] P. Zhou *et al.*, "IEEE 802.11ay-Based mmWave WLANs: Design Challenges and Solutions," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 3, pp. 1654–1681, 2018.
- [14] "TR 37.910: Study on self evaluation towards IMT-2020 submission," 3GPP, Oct. 2019.
- [15] (2015, Sept.) Framework and overall objectives of the future development of IMT for 2020 and beyond. [Online]. Available: <http://www.itu.int/rec/RREC-M.2083>
- [16] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021.
- [17] "Usage statistics of QUIC for websites." [Online]. Available: <https://w3techs.com/technologies/details/ce-quic>
- [18] T. Shreedhar, R. Panda, S. Podanev, and V. Bajpai, "Evaluating QUIC Performance over Web, Cloud Storage and Video Workloads," *IEEE Trans. Netw. Service Manag.*, p. 1, 2021.
- [19] M. Rajiullah *et al.*, "Web Experience in Mobile Networks: Lessons from Two Million Page Visits," in *Proc. TheWebConf.* Association for Computing Machinery, 2019, pp. 1532–1543.
- [20] A. Yu and T. A. Benson, "Dissecting Performance of Production QUIC," in *Proc. TheWebConf.* Association for Computing Machinery, 2021, pp. 1157–1168.
- [21] K. Nepomuceno *et al.*, "QUIC and TCP: A Performance Evaluation," in *Proc. IEEE ISCC*, 2018, pp. 45–51.
- [22] M. Duke, R. Braden, W. Eddy, E. Blanton, and A. Zimmermann, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents," RFC 7414 (Informational), Internet Engineering Task Force, Feb. 2015.
- [23] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over Second (2.5G) and Third (3G) Generation Wireless Networks," RFC 3481 (Best Current Practice), Internet Engineering Task Force, Feb. 2003.
- [24] M. Zhang *et al.*, "Will TCP Work in mmWave 5G Cellular Networks?" *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 65–71, 2019.
- [25] H. D. Le, C. T. Nguyen, V. V. Mai, and A. T. Pham, "On the Throughput Performance of TCP Cubic in Millimeter-Wave Cellular Networks," *IEEE Access*, vol. 7, pp. 178 618–178 630, 2019.
- [26] R. Poorzare and A. C. Auge, "How Sufficient is TCP When Deployed in 5G mmWave Networks Over the Urban Deployment?" *IEEE Access*, vol. 9, pp. 36 342–36 355, 2021.
- [27] L. Ding *et al.*, "Understanding commercial 5G and its implications to (Multipath) TCP," *Comput. Netw.*, vol. 198, p. 108401, 2021.
- [28] G. Sinha, M. R. Kanagarathinam, S. R. Jayaseelan, and G. K. Choudhary, "CQUIC: Cross-Layer QUIC for Next Generation Mobile Networks," in *Proc. IEEE WCNC*, 2020, pp. 1–8.
- [29] H. Wu *et al.*, "Multipath Scheduling for 5G Networks: Evaluation and Outlook," *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 44–50, 2021.
- [30] M. Polese *et al.*, "A Survey on Recent Advances in Transport Layer Protocols," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 4, pp. 3584–3608, 2019.
- [31] Y. Ren *et al.*, "A survey on TCP over mmWave," *Comput. Commun.*, vol. 171, pp. 80–88, 2021.
- [32] R. Poorzare and O. P. Waldhorst, "Toward the Implementation of MPTCP Over mmWave 5G and Beyond: Analysis, Challenges, and Solutions," *IEEE Access*, vol. 11, pp. 19 534–19 566, 2023.
- [33] J. Lorincz, Z. Klarin, and J. Ozegovic, "A Comprehensive Overview of TCP Congestion Control in 5G Networks: Research Challenges and Future Perspectives," *Sensors*, vol. 21, no. 13, 2021.
- [34] S. J. Siddiqi, F. Naeem, S. Khan, K. S. Khan, and M. Tariq, "Towards AI-enabled traffic management in multipath TCP: A survey," *Computer Communications*, vol. 181, pp. 412–427, 2022.
- [35] G. Papastergiou *et al.*, "De-Ossifying the Internet Transport Layer: A Survey and Future Perspectives," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 1, pp. 619–639, 2017.
- [36] J. Postel, "Transmission Control Protocol," RFC 793 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1122, 3168, 6093, 6528.
- [37] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681 (Draft Standard), Internet Engineering Task Force, Sep. 2009.
- [38] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks," RFC 8312 (Informational), Internet Engineering Task Force, Feb. 2018.
- [39] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "QTC: Adaptive Congestion Control with Reinforcement Learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 445–458, 2019.
- [40] M. R. Kanagarathinam *et al.*, "NexGen D-TCP: Next Generation Dynamic TCP Congestion Control Algorithm," *IEEE Access*, vol. 8, pp. 164 482–164 496, 2020.
- [41] W. Na, B. Bae, S. Cho, and N. Kim, "DL-TCP: Deep Learning-Based Transmission Control Protocol for Disaster 5G mmWave Networks," *IEEE Access*, vol. 7, pp. 145 134–145 144, 2019.
- [42] S. Abbasloo, Y. Xu, and H. J. Chao, "C2TCP: A Flexible Cellular TCP to Meet Stringent Delay Requirements," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 918–932, 2019.
- [43] M. Kosek, T. Shreedhar, and V. Bajpai, "Beyond QUIC v1: A First Look at Recent Transport Layer IETF Standardization Efforts," *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 24–29, 2021.
- [44] A. Langley *et al.*, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proc. ACM SIGCOMM*, 2017, pp. 183–196.

- [45] M. Honda *et al.*, “Is It Still Possible to Extend TCP?” in *Proc. ACM SIGCOMM IMC*, 2011, pp. 181–194.
- [46] M. Scharf and S. Kiesel, “Head-of-line Blocking in TCP and SCTP: Analysis and Measurements,” in *Proc. IEEE GLOBECOM*, 2006, pp. 1–5.
- [47] J. Iyengar and I. Swett, “QUIC Loss Detection and Congestion Control,” RFC 9002, May 2021.
- [48] V. Paxson, M. Allman, J. Chu, and M. Sargent, “Computing TCP’s Retransmission Timer,” RFC 6298 (Proposed Standard), Internet Engineering Task Force, Jun. 2011.
- [49] L. Torvalds. Linux kernel. [Online]. Available: <https://github.com/torvalds/linux>
- [50] “Network Simulator 3 (NS-3).” [Online]. Available: <https://www.nsnam.org/>
- [51] M. Mezzavilla *et al.*, “End-to-End Simulation of 5G mmWave Networks,” *IEEE Commun. Surv. Tutor.*, vol. 20, no. 3, pp. 2237–2263, 2018.
- [52] M. R. Akdeniz *et al.*, “Millimeter Wave Channel Modeling and Cellular Capacity Evaluation,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1164–1179, 2014.
- [53] M. Polese, J. M. Jornet, T. Melodia, and M. Zorzi, “Toward End-to-End, Full-Stack 6G Terahertz Networks,” *IEEE Commun. Mag.*, vol. 58, no. 11, pp. 48–54, 2020.
- [54] K. L. Bober *et al.*, “Distributed Multiuser MIMO for LiFi in Industrial Wireless Applications,” *J. Light. Technol.*, p. 1, 2021.
- [55] IEEE 802.11bb: 802.11 Light Communications Amendment - Task Group. [Online]. Available: https://www.ieee802.org/11/Reports/tgbb_update.htm
- [56] I. A. Hemadeh, K. Satyanarayana, M. El-Hajjar, and L. Hanzo, “Millimeter-Wave Communications: Physical Channel Models, Design Considerations, Antenna Constructions, and Link-Budget,” *IEEE Commun. Surv. Tutor.*, vol. 20, no. 2, pp. 870–913, 2018.
- [57] A. V. Alejos, M. G. Sanchez, and I. Cuinas, “Measurement and Analysis of Propagation Mechanisms at 40 GHz: Viability of Site Shielding Forced by Obstacles,” *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3369–3380, 2008.
- [58] C. Anderson and T. Rappaport, “In-building wideband partition loss measurements at 2.5 and 60 GHz,” *IEEE Trans. Wirel. Commun.*, vol. 3, no. 3, pp. 922–928, 2004.
- [59] D. Zhang *et al.*, “Novel Quick Start (QS) method for optimization of TCP,” *Wirel. Netw.*, vol. 22, pp. 211–222, 2016.
- [60] N. Cardwell, S. Savage, and T. Anderson, “Modeling TCP latency,” in *Proc. IEEE INFOCOM*, vol. 3, 2000, pp. 1742–1751.
- [61] D. Zheng, G. Y. Lazarou, and R. Hu, “A stochastic model for short-lived TCP flows,” in *Proc. IEEE ICC*, vol. 1, 2003, pp. 76–81.
- [62] J. Gettys and K. Nichols, “Bufferbloat: Dark Buffers in the Internet,” *Queue*, vol. 9, no. 11, pp. 40–54, Nov. 2011.
- [63] M. Pieska and A. Kassler, “TCP performance over 5G mmWave links — Tradeoff between capacity and latency,” in *Proc. IEEE WiMob*, 2017, pp. 385–394.
- [64] P. J. Mateo, C. Fiandrino, and J. Widmer, “Analysis of TCP Performance in 5G mmWave Mobile Networks,” in *Proc. IEEE ICC*, 2019, pp. 1–7.
- [65] W. Na, D. Lakew, J. Lee, and S. Cho, “Congestion control vs. link failure: TCP behavior in mmWave connected vehicular networks,” *Future Gener. Comput. Syst.*, vol. 101, pp. 1213–1222, July 2019.
- [66] R. Long, Y. C. Liang, Y. Pei, and E. G. Larsson, “Active Reconfigurable Intelligent Surface Aided Wireless Communications,” *IEEE Trans. Wirel. Commun.*, p. 1, 2021.
- [67] C. Liaskos *et al.*, “End-to-End Wireless Path Deployment With Intelligent Surfaces Using Interpretable Neural Networks,” *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6792–6806, 2020.
- [68] “TS 37.340: NR; Multi-connectivity; Overall description; Stage-2,” 3GPP, Mar. 2021.
- [69] C. Pupiales *et al.*, “Multi-Connectivity in Mobile Networks: Challenges and Benefits,” *IEEE Commun. Mag.*, vol. 59, no. 11, pp. 116–122, 2021.
- [70] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, “TCP Extensions for Multipath Operation with Multiple Addresses,” Internet Engineering Task Force, March 2020.
- [71] Y. Liu, Y. Ma, C. Huitema, Q. An, and Z. Li, “Multipath Extension for QUIC,” Internet Engineering Task Force, Internet-Draft draft-liu-multipath-quic-04, Sep. 2021, work in Progress.
- [72] J. Sahoo *et al.*, “A Survey on Replica Server Placement Algorithms for Content Delivery Networks,” *IEEE Commun. Surv. Tutor.*, vol. 19, no. 2, pp. 1002–1026, 2017.
- [73] Y. Mao *et al.*, “A Survey on Mobile Edge Computing: The Communication Perspective,” *IEEE Commun. Surv. Tutor.*, vol. 19, no. 4, pp. 2322–2358, 2017.
- [74] P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture and Computation Offloading,” *IEEE Commun. Surv. Tutor.*, vol. 19, no. 3, pp. 1628–1656, 2017.
- [75] “TS 23.501: System architecture for the 5G System (5GS),” 3GPP, Mar. 2021.
- [76] I. Gerasin and A. Krasilov, “Improving Performance of Web Services in 5G New Radio Systems,” in *Proc. IEEE BlackSeaCom*, 2019, pp. 1–3.
- [77] K. Liu and J. Lee, “On Improving TCP Performance over Mobile Data Networks,” *IEEE Trans. Mob. Comput.*, vol. 15, no. 10, pp. 2522–2536, 2016.
- [78] D. A. Hayes, D. Ros, and O. Alay, “On the importance of TCP splitting proxies for future 5G mmWave communications,” in *IEEE LCN Symposium*, 2019, pp. 108–116.
- [79] M. Drago *et al.*, “QoS Provisioning in 60 GHz Communications by Physical and Transport Layer Coordination,” in *Proc. IEEE MASS*, 2019, pp. 308–316.
- [80] M. Kim *et al.*, “Exploiting Caching for Millimeter-Wave TCP Networks: Gain Analysis and Practical Design,” *IEEE Access*, vol. 6, pp. 69 769–69 781, 2018.
- [81] M. Kim, S. Ko, and S. Kim, “Enhancing TCP end-to-end performance in millimeter-wave communications,” in *Proc. IEEE PIMRC*, 2017, pp. 1–5.
- [82] M. Polese *et al.*, “milliProxy: A TCP proxy architecture for 5G mmWave cellular systems,” in *Proc. Asilomar Conf. Signals Syst. Comput.*, 2017, pp. 951–957.
- [83] S. Floyd, “Limited Slow-Start for TCP with Large Congestion Windows,” RFC 3742 (Experimental), Internet Engineering Task Force, Mar. 2004.
- [84] D. Cavendish *et al.*, “CapStart: An Adaptive TCP Slow Start for High Speed Networks,” in *Proc. INTERNET*, 2009, pp. 15–20.
- [85] S. Ha and I. Rhee, “Taming the Elephants: New TCP Slow Start,” *Comput. Netw.*, vol. 55, no. 9, pp. 2092–2110, June 2011.
- [86] “HyStart++: Modified Slow Start for TCP.” [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-hystartplusplus>
- [87] M. Schapira and K. Winstein, “Congestion-Control Throwdown,” in *Proc. ACM HotNets*, 2017, pp. 122–128.
- [88] R. Adams, “Active Queue Management: A Survey,” *IEEE Commun. Surv. Tutor.*, vol. 15, no. 3, pp. 1425–1476, 2013.
- [89] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive RED: An Algorithm for Increasing the Robustness of RED’s Active Queue Management,” AT&T Center for Internet Research at ICSI, Tech. Rep., 2001.
- [90] K. Zhou, K. Yeung, and V. Li, “Nonlinear RED: A simple yet efficient active queue management scheme,” *Comput. Netw.*, vol. 50, no. 18, pp. 3784–3794, 2006.
- [91] Y. Gong *et al.*, “Fighting the bufferbloat: On the coexistence of AQM and low priority congestion control,” in *Proc. IEEE INFOCOM*, 2013, pp. 3291–3296.
- [92] K. Nichols and V. Jacobson, “Controlling Queue Delay: A Modern AQM is Just One Piece of the Solution to Bufferbloat,” *Queue*, vol. 10, no. 5, pp. 20–34, May 2012.
- [93] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, “Controlled Delay Active Queue Management,” Jan. 2018.
- [94] J. Palmei *et al.*, “Design and Evaluation of COBALT Queue Discipline,” in *Proc. IEEE LANMAN*, 2019, pp. 1–6.
- [95] W. Feng, K. Shin, D. Kandlur, and D. Saha, “The BLUE active queue management algorithms,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 513–528, 2002.
- [96] T. Hoeiland-Joergensen, P. McKenney, D. Taht, A. Zimmermann, J. Gettys, and E. Dumazet, “The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm,” RFC 8290 (Experimental), Internet Engineering Task Force, Jan. 2018.
- [97] M. Shreedhar and G. Varghese, “Efficient fair queuing using deficit round-robin,” *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375–385, 1996.
- [98] M. H. MacGregor and W. Shi, “Deficits for bursty latency-critical flows: DRR+,” in *Proc. IEEE ICON*, 2000, pp. 287–293.

- [99] P. McKeeney, "Stochastic fairness queueing," in *Proc. IEEE INFOCOM*, vol. 2, 1990, pp. 733–740.
- [100] F. Checconi, L. Rizzo, and P. Valente, "QFQ: Efficient Packet Scheduling with Tight Guarantees," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, 2010.
- [101] Y. Qian, Z. Lu, and Q. Dou, "QoS scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin," in *Proc. IEEE ICCD*, 2010, pp. 52–59.
- [102] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.
- [103] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 675–689, 1997.
- [104] T. Azzino *et al.*, "X-TCP: a cross layer approach for TCP uplink flows in mmwave networks," in *Proc. Med-Hoc-Net Workshop*, 2017, pp. 1–6.
- [105] F. Lu *et al.*, "CQIC: Revisiting Cross-Layer Congestion Control for Cellular Networks," in *Proc. HotMobile*, 2015, pp. 45–50.
- [106] Y. Liu *et al.*, "BESS: BDP Estimation Based Slow Start Algorithm for MPTCP in mmWave-LTE Networks," in *Proc. IEEE VTC*, 2018, pp. 1–5.
- [107] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168 (Proposed Standard), Internet Engineering Task Force, Sep. 2001, updated by RFCs 4301, 6040.
- [108] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, "Quick-Start for TCP and IP," RFC 4782 (Experimental), Internet Engineering Task Force, Jan. 2007.
- [109] I. F. Akyildiz *et al.*, "xStream: A New Platform Enabling Communication Between Applications and the 5G Network," in *Proc. IEEE GC Wshps*, 2018, pp. 1–6.
- [110] H. Iwasawa, K. Tokunaga, and N. Takaya, "Available-Bandwidth Information Based TCP Congestion Control Algorithm on Multi-RAT Networks," in *Proc. IEEE GCC*, 2017, pp. 1–6.
- [111] V. Poirot, M. Ericson, M. Nordberg, and K. Andersson, "Energy efficient multi-connectivity algorithms for ultra-dense 5G networks," *Wirel. Netw.*, vol. 26, pp. 2207–2222, 2020.
- [112] M. Suer, C. Thein, H. Tchouankem, and L. Wolf, "Multi-Connectivity as an Enabler for Reliable Low Latency Communications—An Overview," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 1, pp. 156–169, 2020.
- [113] C. Raiciu, M. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," RFC 6356 (Experimental), Internet Engineering Task Force, Oct. 2011.
- [114] R. Khalili, N. Gast, M. Popovic, and J. Le Boudec, "MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1651–1665, 2013.
- [115] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, Design, and Implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, 2016.
- [116] (2021, Aug.) MultiPath TCP - Linux Kernel implementation. [Online]. Available: <https://www.multipath-tcp.org/>
- [117] M. Polese, R. Jana, and M. Zorzi, "TCP in 5G mmWave networks: Link level retransmissions and MP-TCP," in *Proc. IEEE INFOCOM WKSHPS*, 2017, pp. 343–348.
- [118] M. P. McGarry, R. Shakya, M. I. Ohannessian, and R. Ferzli, "Optimal Caching Router Placement for Reduction in Retransmission Delay," in *Proc. ICCCN*, 2011, pp. 1–8.
- [119] W. Wong, M. Giralidi, M. F. Magalhaes, and J. Kangasharju, "Content Routers: Fetching Data on Network Path," in *Proc. IEEE ICC*, 2011, pp. 1–6.
- [120] "Network Caching White Paper," Cisco, 2008. [Online]. Available: https://www.cisco.com/c/dam/global/de_at/assets/docs/Net_Caching.pdf
- [121] S. Safavat, N. Sapavath, and D. Rawat, "Recent advances in mobile edge computing and content caching," *Digit. Commun. Netw.*, vol. 6, no. 2, pp. 189–194, 2020.
- [122] S. Li, J. Xu, M. van der Schaar, and W. Li, "Popularity-driven content caching," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.
- [123] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid Content Caching in 5G Wireless Networks: Cloud Versus Edge Caching," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 5, pp. 3030–3045, 2018.
- [124] F. Spinelli and V. Mancuso, "Toward Enabled Industrial Verticals in 5G: A Survey on MEC-Based Approaches to Provisioning and Flexibility," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 1, pp. 596–630, 2021.
- [125] A. Belogaeu *et al.*, "Cost-Effective V2X Task Offloading in MEC-Assisted Intelligent Transportation Systems," *IEEE Access*, vol. 8, pp. 169 010–169 023, 2020.
- [126] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC 7230 (Proposed Standard), Internet Engineering Task Force, Jun. 2014.
- [127] D. Moltchanov *et al.*, "Analytical TCP Model for Millimeter-Wave 5G NR Systems in Dynamic Human Body Blockage Environment," *Sensors*, vol. 20, no. 14, 2020.
- [128] K. Leung and V. Li, "Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges," *IEEE Commun. Surv. Tutor.*, vol. 8, no. 4, pp. 64–79, 2006.
- [129] B. Sardar and D. Saha, "A survey of TCP enhancements for last-hop wireless networks," *IEEE Commun. Surv. Tutor.*, vol. 8, no. 3, pp. 20–34, 2006.
- [130] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135 (Informational), Internet Engineering Task Force, Jun. 2001.
- [131] M. Zhang *et al.*, "TCP dynamics over mmwave links," in *Proc. IEEE SPAWC*, 2017, pp. 1–6.
- [132] —, "The Bufferbloat Problem over Intermittent Multi-Gbps mmWave Links," *Computing Research Repository*, 2016.
- [133] C. Chaccour *et al.*, "Seven Defining Features of Terahertz (THz) Wireless Systems: A Fellowship of Communication and Sensing," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 2, pp. 967–993, 2022.
- [134] R. Wang *et al.*, "TCP startup performance in large bandwidth networks," in *Proc. IEEE INFOCOM*, vol. 2, 2004, pp. 796–805.
- [135] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582 (Proposed Standard), Internet Engineering Task Force, Apr. 2012.
- [136] S. Floyd, "HighSpeed TCP for Large Congestion Windows," RFC 3649 (Experimental), Internet Engineering Task Force, Dec. 2003.
- [137] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proc. IEEE INFOCOM*, vol. 4, 2004, pp. 2514–2524.
- [138] T. Kozu, Y. Akiyama, and S. Yamaguchi, "Improving RTT Fairness on CUBIC TCP," in *Proc. CANDAR*, 2013, pp. 162–167.
- [139] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *Proc. ACM SIGCOMM CCR*, vol. 24, no. 4, Oct. 1994.
- [140] K. N. Sriji, L. Jacob, and A. L. Ananda, "TCP Vegas-A: solving the fairness and rerouting issues of TCP Vegas," in *Proc. IEEE IPCCC*, 2003, pp. 309–316.
- [141] W. Zhou, W. Xing, Y. Wang, and J. Zhang, "TCP Vegas-V: Improving the performance of TCP Vegas," in *Proc. ACAI*, 2012, pp. 2034–2039.
- [142] J. Sing and B. Soh, "TCP New Vegas: Performance Evaluation and Validation," in *Proc. IEEE ISCC*, 2006, pp. 541–546.
- [143] Y. Zaki *et al.*, "Adaptive Congestion Control for Unpredictable Cellular Networks," *Proc. ACM SIGCOMM CCR*, vol. 45, no. 4, 2015.
- [144] P. Goyal *et al.*, "Elasticity Detection: A Building Block for Delay-Sensitive Congestion Control," in *Proc. ANRW*, 2018, p. 75.
- [145] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)," RFC 6817 (Experimental), Internet Engineering Task Force, Dec. 2012.
- [146] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-Speed and Long Distance Networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.
- [147] S. Liu, T. Basar, and R. Srikant, "TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks," *Perform. Evaluation*, vol. 65, no. 6, pp. 417–440, 2008.
- [148] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, pp. 216–228, 2003.
- [149] A. Baiocchi, A. P. Castellani, and F. Vacirca, "YeAH-TCP: Yet Another Highspeed TCP," in *Proc. PFLDnet*, 2007.
- [150] S. Mascolo *et al.*, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *Proc. MobiCom*, 2001, pp. 287–297.

- [151] R. Ferorelli *et al.*, "Live Internet measurements using Westwood+ TCP congestion control," in *Proc. IEEE GLOBECOM*, vol. 3, 2002, pp. 2583–2587.
- [152] L. A. Grieco and S. Mascolo, "Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control," *Proc. ACM SIGCOMM CCR*, vol. 34, no. 2, pp. 25–38, 2004.
- [153] N. Cardwell *et al.*, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, pp. 20–53, 2016.
- [154] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *Proc. IEEE ICNP*, 2017, pp. 1–10.
- [155] K. Miyazawa, K. Sasaki, N. Oda, and S. Yamaguchi, "Cyclic Performance Fluctuation of TCP BBR," in *Proc. IEEE COMPSAC*, vol. 1, 2018, pp. 811–812.
- [156] P. Farrow, "Performance analysis of heterogeneous TCP congestion control environments," in *Proc. IFIP/IEEE PEMWN*, 2017, pp. 1–6.
- [157] E. Atxutegi *et al.*, "On the Use of TCP BBR in Cellular Networks," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 172–179, 2018.
- [158] F. Li, J. W. Chung, X. Jiang, and C. Mark, "TCP CUBIC versus BBR on the Highway," in *Proc. PAM*, 2018, pp. 1–12.
- [159] N. Cardwell *et al.*, "BBR v2: A model-based congestion control," in *Present. in ICCRG at IETF 104th meeting*, 2019. [Online]. Available: https://lafibre.info/testdebut/linux/201903_bbr_v2_doc_ietf104.pdf
- [160] X. Nie *et al.*, "Dynamic TCP Initial Windows and Congestion Control Schemes Through Reinforcement Learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1231–1247, 2019.
- [161] W. Li *et al.*, "SmartCC: A Reinforcement Learning Approach for Multipath TCP Congestion Control in Heterogeneous Networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2621–2633, 2019.
- [162] T. Zhang and S. Mao, "Machine Learning for End-to-End Congestion Control," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 52–57, 2020.
- [163] A. Srivastava, F. Fund, and S. S. Panwar, "An Experimental Evaluation of Low Latency Congestion Control for mmWave Links," in *Proc. IEEE INFOCOM WKSHPS*, 2020, pp. 352–357.
- [164] L. Diez *et al.*, "Can We Exploit Machine Learning to Predict Congestion over mmWave 5G Channels?" *Appl. Sci.*, vol. 10, no. 18, 2020.
- [165] R. Poorzare and A. C. Auge, "FB-TCP: A 5G mmWave Friendly TCP for Urban Deployments," *IEEE Access*, vol. 9, pp. 82 812–82 832, 2021.
- [166] L. Diez *et al.*, "Learning congestion over millimeter-wave channels," in *Proc. IEEE WiMob*, 2020, pp. 1–6.
- [167] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.
- [168] M. Pieska, A. J. Kessler, H. Lundqvist, and T. Cai, "Improving TCP Fairness over Latency Controlled 5G mmWave Communication Links," in *Proc. WSA*, 2018, pp. 1–8.
- [169] M. Zhang *et al.*, "Transport layer performance in 5G mmWave cellular," in *Proc. IEEE INFOCOM WKSHPS*, 2016, pp. 730–735.
- [170] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567, Jul. 2015.
- [171] M. Ihlar, A. Nazari, and R. Skog, "Low latency, high flexibility - Virtual AQM, Ericson Technology Review." [Online]. Available: <https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/virtual-aqm-for-mobile-networks>
- [172] J. H. Salim and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks," RFC 2884 (Informational), Internet Engineering Task Force, Jul. 2000.
- [173] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proc. IEEE INFOCOM*, vol. 3, 2000, pp. 1157–1165.
- [174] M. Scharf, S. Hauger, and J. Kogel, "Quick-Start TCP: From Theory to Practice," *Proc. PFLDnet*, Mar. 2008.
- [175] "TS 23.402: Architecture enhancements for non-3GPP accesses," 3GPP, Mar. 2016.
- [176] G. MacCartney and T. Rappaport, "Millimeter-Wave Base Station Diversity for 5G Coordinated Multipoint (CoMP) Applications," *IEEE Trans. Wirel. Commun.*, vol. 18, no. 7, pp. 3395–3410, 2019.
- [177] Ericsson. (2021, June) What, Why and How: the Power of 5G Carrier Aggregation. [Online]. Available: <https://www.ericsson.com/en/blog/2021/6/what-why-how-5g-carrier-aggregation>
- [178] A. Aijaz, "Packet Duplication in Dual Connectivity Enabled 5G Wireless Networks: Overview and Challenges," *IEEE Commun. Stand. Mag.*, vol. 3, no. 3, pp. 20–28, 2019.
- [179] S. Kang, S. Choi, G. Lee, and S. Bahk, "A Dual-Connection Based Handover Scheme for Ultra-Dense Millimeter-Wave Cellular Networks," in *Proc. IEEE GCC*, 2019, pp. 1–6.
- [180] F. Hu *et al.*, "Cellular-Connected Wireless Virtual Reality: Requirements, Challenges, and Solutions," *IEEE Commun. Mag.*, vol. 58, no. 5, pp. 105–111, 2020.
- [181] D. Michalopoulos and V. Pauli, "Data Duplication for High Reliability: A Protocol-Level Simulation Assessment," in *Proc. IEEE ICC*, 2019, pp. 1–7.
- [182] L. Weedage, C. Stegehuis, and S. Bayhan, "Impact of Multi-connectivity on Channel Capacity and Outage Probability in Wireless Networks," *IEEE Trans. Veh. Technol.*, pp. 1–14, 2023.
- [183] T. Mumtaz, S. Muhammad, M. Aslam, and N. Mohammad, "Dual Connectivity-Based Mobility Management and Data Split Mechanism in 4G/5G Cellular Networks," *IEEE Access*, vol. 8, pp. 86 495–86 509, 2020.
- [184] M. Suslopárov, A. Krasilov, and E. Khorov, "Providing High Capacity for AR/VR traffic in 5G Systems with Multi-Connectivity," in *Proc. IEEE BlackSeaCom*, 2022, pp. 385–390.
- [185] M. Drago *et al.*, "Reliable Video Streaming over mmWave with Multi Connectivity and Network Coding," in *Proc. ICNC*, 2018, pp. 508–512.
- [186] E. Dias *et al.* (2022, 05) Millimeter-Wave in Milliseconds: Sliding Window Network Coding Outperforms Rateless Codes. [Online]. Available: <https://arxiv.org/abs/2205.00793>
- [187] M. S. Karim, A. Douik, P. Sadeghi, and S. Sorour, "On Using Dual Interfaces With Network Coding for Delivery Delay Reduction," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 6, pp. 3981–3995, 2017.
- [188] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824 (Experimental), Internet Engineering Task Force, Jan. 2013.
- [189] C. Paasch and O. Bonaventure, "Multipath TCP," *Communications of the ACM*, vol. 57, no. 4, pp. 51–57, Apr. 2014.
- [190] Y. Liu *et al.* Multipath Extension for QUIC. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/>
- [191] S. Ferlin, T. Dreiholz, and O. Alay, "Multi-path transport over heterogeneous wireless networks: Does it really pay off?" in *Proc. IEEE GCC*, 2014, pp. 4807–4813.
- [192] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental Evaluation of Multipath TCP Schedulers," in *Proc. ACM SIGCOMM CSWS*. Association for Computing Machinery, 2014, pp. 27–32.
- [193] F. Yang, Q. Wang, and P. D. Amer, "Out-of-Order Transmission for In-Order Arrival Scheduling for Multipath TCP," in *Proc. AINA*, 2014, pp. 749–752.
- [194] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proc. IFIP Networking and Workshops*, 2016, pp. 431–439.
- [195] Y. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths," *SIGMETRICS Performance Evaluation Rev.*, vol. 45, no. 1, pp. 33–34, 2017.
- [196] P. Hurtig *et al.*, "Low-Latency Scheduling in MPTCP," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 302–315, 2019.
- [197] N. Kuhn *et al.*, "DAPS: Intelligent delay-aware packet scheduling for multipath transport," in *Proc. IEEE ICC*, 2014, pp. 1222–1227.
- [198] J. Wang, J. Liao, and T. Li, "OSIA: Out-of-Order Scheduling for In-Order Arriving in Concurrent Multi-Path Transfer," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 633–643, Mar. 2012.
- [199] H. Zhang *et al.*, "ReLeS: A Neural Adaptive Multipath Scheduler based on Deep Reinforcement Learning," in *Proc. IEEE INFOCOM*, 2019, pp. 1648–1656.
- [200] H. Wu *et al.*, "Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2295–2310, 2020.