An Empirical Study of 5G: Effect of Edge on Transport Protocol and Application Performance

Hyoyoung Lim, *Student Member, IEEE*, Jinsung Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Junseon Kim, Anh Nguyen, Kwang Taik Kim, *Senior Member, IEEE*, Youngbin Im, *Member, IEEE*, Mung Chiang, *Fellow, IEEE*, Dirk Grunwald, *Senior Member, IEEE*, Kyunghan Lee, *Member, IEEE*, and Sangtae Ha, *Senior Member, IEEE*

***Abstract*— In this paper, we conduct a measurement study on operational 5G networks deployed across different frequency bands (mmWave and sub-6GHz) and server locations (mobile edge and Internet cloud). Specifically, we assess 5G performance in both uplink and downlink across multiple operators' networks. We then carry out extensive comparisons of transport-layer protocols using ten different algorithms in full-fledged 5G networks, including an edge computing environment. Finally, we evaluate representative mobile applications over the 5G network with and without edge servers. Our comprehensive measurements provide several insights that affect the experience of 5G users: (i) With a 5G edge server, existing TCP congestion control algorithms can achieve throughput up to 1.8Gbps with only a single flow. (ii) The maximum TCP receive buffer size, which is set by off-the-shelf 5G phones, can limit the throughput performance of 5G networks, which is not observed in 4G LTE-A networks. (iii) Despite significant latency gains in download-centric applications, the 5G edge service provides limited benefits to CPU-intensive tasks or those that use significant uplink bandwidth. To our knowledge, this is the first measurement-driven understanding of 5G edge computing "in the wild," which can provide an answer to how edge computing would perform in real 5G networks.**

***Index Terms*—5G, network measurement, 5G coverage, edge computing, transport protocol, network latency, congestion control.**

Manuscript received 16 August 2022; revised 23 March 2023; accepted 24 April 2023. Date of publication 12 May 2023; date of current version 6 March 2024. This research was supported in part by the IEDC under Grant A293-2-21Fund-10245 and in part by the IITP under Grants 2022-0-00420 and IITP-2023-2021-0-01817 funded by the Korea government (MSIT). Recommended for acceptance by E. Hossain. *(Corresponding author: Jinsung Lee.)*

Hyoyoung Lim, Jinsung Lee, Sandesh Dhawaskar Sathyanarayana, Dirk Grunwald, and Sangtae Ha are with the Department of Computer Science, University of Colorado Boulder, Boulder, CO 80027 USA (e-mail: hyoyoung.lim@colorado.edu; jinsung.lee@colorado.edu; sadh0344@colorado.edu; dirk.grunwald@colorado.edu; sangtae.ha@colorado.edu).

Anh Nguyen is with the Department of Computer Science, University of Montana, Missoula, MT 59812 USA (e-mail: anh.nguyen@umontana.edu).

Jongyun Lee, Junseon Kim, and Kyunghan Lee are with the Department of Computer Science, Seoul National University, Seoul 08826, Republic of Korea (e-mail: jongyunlee@snu.ac.kr; junseonkim@snu.ac.kr; kyunghanlee@snu.ac.kr).

Youngbin Im is with the Department of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology, Ulsan 44919, Republic of Korea (e-mail: ybim@unist.ac.kr).

Kwang Taik Kim and Mung Chiang are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: kimkt@purdue.edu; chiang@purdue.edu).

Digital Object Identifier 10.1109/TMC.2023.3274708

I. Introduction

ACCORDING to GSA (Global Mobile Suppliers Association)'s market survey [1], 153 operators have launched commercial 5G networks in 64 countries as of March 2021. Most 5G operators currently rely on sub-6 GHz frequency bands while some are deploying mmWave-based 5G new radio (5G NR), which can unleash the full potential of 5G despite coverage issues. Almost all operators currently conform to the non-standalone (NSA) architecture in which the 5G radio access network (5G RAN) is connected to a 4G LTE core network [2].

Edge computing is an emerging architectural paradigm that provides ample computing, storage, and networking resources within the cellular core in close proximity to mobile users. This paradigm alleviates the backhaul traffic in the cellular core and helps to realize ultra-low latency, high bandwidth, and agile mobile services. Compared to traditional cellular business models, it is expected to boost a tight collaboration between cellular operators and third-party application/content providers by deploying applications/content on the mobile edge. However, there has been no measurement-based research that assesses the real value of edge computing in commercial 5G networks due to their black-boxed design, motivating our research.

In this paper, for the first time, we carry out comprehensive measurements to evaluate the real potential of edge computing in operational 5G networks. As a fundamental yet crucial step, we investigate the networking perspective of using edge servers by evaluating the performance of different TCP congestion control algorithms (CCAs) and popular mobile applications. Within the scope of this paper, we focus on TCP because it has become the dominant transport-layer protocol in cellular networks due to the popularity of HTTP-based applications [3], [4]. Furthermore, to understand the impact of several deployment choices such as frequency spectrum and location dependency in 5G, we perform in-depth comparisons between commercial 5G networks deployed in the U.S. and South Korea: the former has deployed 5G as small cells on the mmWave frequency bands (e.g., 24–39 GHz), and the latter has deployed nation-wide coverage on sub-6 GHz bands (e.g., 3.4–3.6 GHz), detailed in Section II-A. Our experiments span more than six months and have consumed over 20.9 TB of 5G/4G cellular data.

From the transport-layer perspective, there has been a limited number of preliminary studies on TCP performance over 5G cellular networks [5], [6], [7], [8]. Table I summarizes a comparison between their work and ours. In particular, Narayanan et

TABLE I
QUALITATIVE COMPARISONS WITH PRIOR WORKS ON 5G PERFORMANCE MEASUREMENTS

Property	[5]	[6]	[7]	[8]	[9]	[10]	Our work
Year	2017	2019	2020	2020	2021	2021	
Type of 5G frequency	Simulation -based	Simulation -based	mmWave (US)	Sub-6GHz (CN)	mmWave (US)	Sub-6GHz (CN)	mmWave (US), sub-6GHz (KOR)
Throughput and latency	✓	✓	✓	✓	–	✓	✓
Stationary UE	–	–	✓	–	–	–	✓
Mobility UE	–	✓	✓	–	✓	–	Car, Subway
Coverage	–	–	–	✓	–	–	✓
Application performance	–	–	✓	✓	–	✓	✓
Energy consumption	–	–	–	✓	✓	–	–
TCP congestion controls	2 CCAs	4 CCAs	–	–	1 CCA	–	10 different CCAs
Edge performance	–	✓	–	–	–	Off-site edge	On-site edge

al. [7] evaluated the performance of parallel TCP connections in various 5G environments in the U.S. while fixing the default CCA. They found that a single TCP flow cannot achieve the maximum throughput (tput), which is not consistent from our measurement. Zhang et al. [8] compared different CCAs in China's sub-6 GHz 5G networks and showed that loss-based CCAs could not achieve maximum throughput due to severe packet losses in the wired network part. However, we find that this observation is not applicable to operational 5G networks deployed in the U.S. and South Korea. In our study, we provide a detailed comparison of different TCP CCAs in full-fledged 5G networks and assess them under practical scenarios such as cellular edge, obstruction, and vehicular mobility, which can be crucial to the user's experience. *To our knowledge, this is the first in-depth study of different CCAs over commercial 5G networks with different frequency bands (i.e., sub-6 GHz and mmWave) and operational 5G edge computing environments deployed in the providers' core networks.*

Key Findings and Contributions: Our measurement campaign provides several insights that can demystify the deployed 5G networks, which are summarized as follows:

(i) We find that with a 5G edge server, existing transport-layer CCAs can achieve throughput up to 1.8 Gbps (Section V-A and Fig. 5(a)), which is the maximum achievable bandwidth with the setting of devices we used. However, only low-latency CCAs can maintain sub-10 ms round-trip times (RTTs). We also find that this performance characteristic is consistent with a 4G edge server, as shown in Fig. 5(d). More importantly, even in the edge environment, we see the bufferbloat of up to 6× by loss-based CCAs.

(ii) The maximum TCP receive buffer size, which is typically fixed, plays a critical role in the achieved throughput in 5G irrespective of different CCAs (Fig. 5(b)). In other words, the performance is limited by the maximum buffer size. Particularly, a 2 Gbps link with 32 ms latency has a bandwidth-delay product (BDP) of 8 MB, matching the maximum TCP receive buffer size parameter (`tcp_rmem_max`) in Android 10, 11 and 12. This also means that the latency above 32 ms would not achieve the maximum throughput. However, we do not observe this limitation in the case of 4G LTE-A, as its receive buffer size (e.g., 4 MB) is sufficient for its speed.

(iii) Our evaluation of transport-layer protocols shows that applications need to choose a CCA carefully to achieve both gigabit throughput and sub-10 ms latency over 5G networks and beyond, suggesting an application-level knob for selecting the transport-layer CCA. Based on our comparison, we recommend using `ExLL` [11] for applications sensitive to both throughput and latency. Overall, Google's `BBR` [12] performs well in 5G, but the jitter needs to be taken into account under a strict application requirement.

(iv) We observe a wide range of performance degradation due to severe penetration loss of up to 30 dB (1000× reduction) in received signal strength with operational mmWave 5G links (Section VI and Fig. 8). When penetration loss accompanies severe packet losses, we see a considerable decrease in the congestion window (CWND)[1] as well as the slow start threshold, incurring a sluggish increase in CWND. This causes a significant delay in restoring full bandwidth (e.g., 31 seconds in the default `CUBIC` [13]).

(v) We experimentally confirm that several mobile applications benefit from using an edge computing server deployed in the vicinity of a 5G cell tower (Section VII). However, despite a considerable latency gain in download-centric applications, edge computing provides limited benefits to CPU-intensive tasks or those using significant uplink bandwidth.

II. BACKGROUND

A. 5G Network

Frequency Bands. Frequency bands for 5G NR are separated into two different frequency ranges [14]. Frequency Range 1 (FR1) includes sub-6 GHz frequency bands, some of which are traditionally used by previous standards but have been extended to cover potential new spectrum offerings from 410 MHz to 7,125 MHz. Frequency Range 2 (FR2), on the other hand, includes *millimeter-wave* bands from 24.25 GHz to 52.6 GHz, which have shorter coverage but higher bandwidth than those in FR1.

Deployment Options: 5G has five different deployment options, where standalone (SA) options consist of only one generation of radio access technology, and non-standalone (NSA) options include two generations of radio access technologies (i.e., 4G LTE and 5G) [15]. Among them, NSA option 3 is the most popular as of now; the RAN is composed of 4G's evolved NodeBs (eNBs) as the *master* node and 5G's next-generation

[1]The CWND represents the sending rate in the number of packets.

NodeBs (gNBs) as the *secondary* node,[2] where eNB and gNB are connected to a 4G LTE's Evolved Packet Core (EPC). We empirically find that in operational 5G, the gNB's user plane connection for data traffic is established directly to the EPC and all gNBs are co-located with eNBs as in [7], [8].

Commercialization Status: South Korea's three mobile carriers and Verizon in the U.S. launched the world's first commercial 5G services on April 11, 2019 [18], [19]. Since then, GSMA reported that 106 commercial 5G networks had been launched as of September 2020 [20]. In the U.S., Verizon and AT&T have deployed 5G services using mmWave bands,[3] while in South Korea, major operators have started 5G service using sub-6 GHz mid-bands [23], [24].

B. Transport-Layer Congestion Control

In our study, we considered CCAs at the sender. Testing the receiver-side CCAs would require changes to Android kernel. However, at the time of measurement, rooting was not available for the 5G phones we tested.

Loss-Based Window Control: This type of CCA (e.g., Reno and CUBIC) uses a packet loss as a congestion signal to reduce its sending rate in the network. However, continuously increasing the sending rate until a packet loss happens leads to excessive buffering at the bottleneck link, called *bufferbloat* [25]. We evaluated `CUBIC` [13], which is the default TCP CCA in Linux (hence, the default in most Android devices) and in Windows [26]. It modifies the linear window growth function of conventional TCP standards to be a cubic function to improve TCP's scalability over fast and long-distance networks. During a steady-state, `CUBIC` increases the CWND aggressively when the CWND is far from the saturation point and slowly when it is close to the saturation point.

Delay-Based Window Control: Delay-based CCAs exploit packet delays (either one-way or RTT) as a congestion signal rather than packet loss. In this category, we evaluated `Copa` [27], `ExLL` [11], `Sprout` [28], `Vegas` [29], and `Verus` [30]. Those CCAs are all of which use packet delays in a different manner. For example, TCP `Vegas` measures the difference δ between expected throughput and actual throughput based on RTTs. When δ is less than a low threshold of α, `Vegas` believes the path is not congested and thus increases the sending rate. When δ is larger than an upper threshold β, which is a strong indication of congestion, `Vegas` reduces the sending rate. Otherwise, `Vegas` maintains the current sending rate. The expected throughput is calculated by dividing the current CWND by the minimum RTT, which likely indicates the delay when the path is not congested. For each RTT, `Vegas` computes the actual throughput by dividing the number of packets sent by the sampled RTT.

On the other hand, the sender-side `ExLL` estimates the mobile client's receiving rate by calculating max throughput estimate (MTE_S) from ACKs received at the sender as $\text{MTE}_S = \text{CWND}/\Delta t$, where Δt is the time difference between the first ACK arrival and the last ACK arrival for the group of packets sent as CWND. It exploits FAST's equation-based CWND adaptation, which considers the ratio between the measured RTT and min RTT while persistently pushing CWND to grow by a constant value [31]. Therefore, `ExLL` provides proportional fairness that does not penalize flows with large propagation delays and guarantee convergence speed and stability in a dynamic network.

Rate-Based Window Control: Several researchers have proposed smoothing traffic bursts by evenly spacing between data transmissions over each RTT [32]. We evaluated Google's `BBR` [12]. `BBR` is designed to exploit the pacing [32] technique and maintains two key variables: bottleneck bandwidth (BtlBW) and round-trip propagation time (RTprop). `BBR` switches between different states of its state machine based on the observed BtlBw, RTprop, and the number of packets in flight. `BBR` periodically enters the ProbeRTT state to reduce its CWND and drains the queue to reset itself.

Cellular-Specific Window Control: These algorithms are designed to operate in existing cellular networks with highly fluctuating wireless channel conditions. In this category, we evaluated `Sprout`, `Verus`, `ExLL`, and `C2TCP` [33], some of which are also included in the delay-based window control algorithms. Among them, `Verus` uses delay measurements to react quickly to cellular networks' capacity changes without explicitly predicting the cellular channel dynamics. The key idea of `Verus` is to continuously learn a delay profile that captures the relationship between the end-to-end packet delay and sending window size without causing congestion over short epochs (i.e., 5 ms). It then uses this relationship to increase or decrease the window size based on the observed short-term packet delay variations.

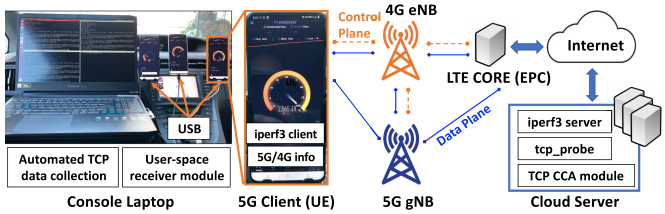


Fig. 1. Experimental setup for our measurements.

III. Measurement Methodology

Due to the difficulty of accessing the internal cellular systems, we take an end-to-end approach. Specifically, we collect transport-layer statistics at both endpoints and cellular information at the 5G phone for detailed analysis, as shown in Fig. 1.

A. Measurement Setup

Server Setup: We use servers from CloudLab [34], which provides high performance computing and networks across several states in the U.S. The average ping latency between our 5G phones and the servers in the U.S. ranged from 22 ms to 70 ms, depending on the server's location. To reflect the distribution of commercial cloud computing resources and highlight the *range* of expected performance, we used servers in multiple locations. Each server machine has two Intel 8-core CPUs at 2.4 GHz, 10 Gbps Ethernet, and 128 GB memory, and runs Ubuntu 16.04 with Linux kernel 4.13.1. To check whether the wireline between the servers and EPC's gateway is the bottleneck in the end-to-end path, we first measure the maximum achievable throughput on the 5G link using Speedtest, which typically connects to a nearby

[2]This is also called E-UTRA-NR Dual Connectivity (EN-DC) [16], [17].

[3]Major U.S. operators have also deployed 5G at sub-6 GHz bands very recently [21], [22], but at the time of our measurement, we could access only mmWave 5G in our tested locations.

server with sub-10 ms RTT [35]. Then, we exploit iperf3 [36] to estimate the maximum available throughput between the cloud server and the 5G phone. Our iperf3 UDP results were comparable to Speedtest's, indicating that the wireline was not the bottleneck. We also set up servers from a commercial cloud service in South Korea in a similar manner.

Client Setup: Our measurement involves three 5G phone models as user equipments (UEs): Samsung Galaxy S20 Plus/Ultra (Qualcomm Snapdragon 865 [37]), Galaxy S10 (Snapdragon 855 [38]), and Galaxy Note 10 Plus (Samsung Exynos 9825 [39]). The former phones, Samsung Galaxy S20 Plus/Ultra, are used in mmWave 5G networks of AT&T and Verizon, while the latter, Galaxy S10 and Galaxy Note 10 Plus, are used in South Korea's 5G networks at sub-6 GHz bands. All phones run Android 10 with Linux kernel 4.19.87. We use TCP CCAs implemented in the Linux kernel at the server, and the 5G phones are used as receivers to measure TCP and application performance over 5G/4G LTE-A networks. For cellular-specific CCAs that are implemented in the user space, such as `Verus`, we cross-compiled their client applications using Android NDK toolset [40] to make them executable on smartphones.

Setting for High Performance: To obtain the maximum TCP performance, we tune several parameters related to TCP performance in Linux [41]. We set the `tcp_wmem` parameter with a considerable value (e.g., 64 MB) at the TCP sender (i.e., edge and cloud servers), which denotes the amount of memory reserved for send buffer for a TCP socket. We disable `tcp_no_metrics_save` to not cache some TCP parameters (e.g., slow start threshold) during repeated experiments. For `CUBIC`, we also disable `HyStart` to make it fair with other loss-based mechanisms in the slow start phase.

B. Data Collection and Processing

Tools for Measurement: We use iperf3 for TCP/UDP traffic generation. We customized `tcp_info.c` and cross-compiled iperf3 to obtain detailed TCP information. To further collect internal TCP variables in fine granularity, we installed the `tcp_probe` module by modifying `tcp_probe.c`.

To capture 5G/4G LTE-related information in the Android OS, we use the `dumpsys` command to collect information every 200 ms during the test. In particular, we use: (i) `CellIdentityLte` to obtain Cell ID, (ii) `CellSignalStrengthNr` to obtain 5G NR signal strength related information when the UE is connected to 5G NR, and (iii) `CellSingal StrengthLte` to obtain LTE signal strength related information. Note that the measurement tools we have developed and the dataset can be downloaded from our repository [42].

Metrics and Variables for TCP Tracing: We collect the achieved throughput[4] and RTT along with their variations. To ensure a fair comparison and reduce the impact of initialization phases required by some schemes, we exclude the first 10 seconds of each experiment. We also check the slow start performance during the initialization phases. Furthermore, we keep track of each CCA's detailed behavior by tracing several key parameters such as CWND, advertised receive window (RWND), number of in-flight packets, packet loss, and smoothed RTT with variations. We also monitor algorithm-specific variables if needed.

[4]An instant throughput is measured every 200 ms interval.

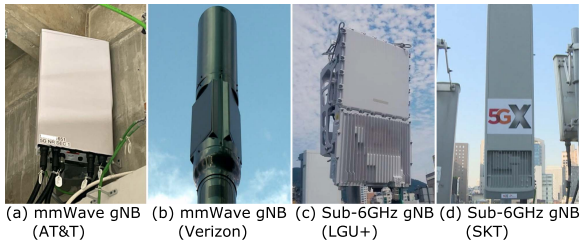


Fig. 2. Deployed gNBs for operational 5G networks.

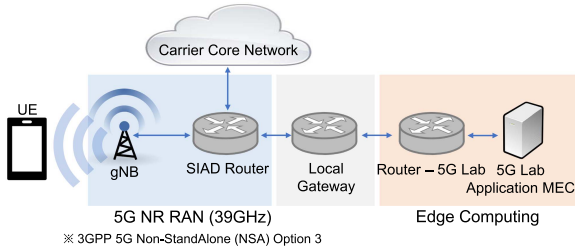


Fig. 3. mmWave 5G Testbed for Edge Services.

Logistics and Post-Processing: Before each experiment, we check the maximum throughput with Speedtest [35], the minimum RTT with ping, and the end-to-end path using traceroute. For stationary scenarios, we run iperf3 with different CCAs for 70 seconds each. However, for mobility cases, we run iperf3 for more than five minutes to ensure a sufficient number of handovers in a single experiment.

Test Locations: To understand the impact of realistic radio propagation over operational 5G networks, we select multiple urban locations based on the operators' 5G coverage maps [43], [44]. Our measurements were conducted in three different locations for Verizon's 5G in the US: (i) a community park measuring 0.58 km $\times$ 0.23 km, (ii) a residential area measuring 0.55 km $\times$ 0.43 km, and (iii) a business complex area measuring 0.62 km $\times$ 0.47 km. For sub-6 GHz 5G in South Korea, we conducted tests in three places in a densely populated city: (i) a campus measuring 0.43 km $\times$ 0.73 km, (ii) a downtown area measuring 0.65 km $\times$ 0.38 km, and (iii) a subway route with a length of 1.5 km. Fig. 2 shows a snapshot of the deployed 5G gNBs used in our measurements.

C. mmWave 5G Testbed for Edge Services

We utilize a comprehensive end-to-end 5G network testbed shown in (Fig. 3) located on a university campus in the U.S. This indoor testbed occupies an area of approximately 150 m^2 and comprises commercial 5G gNBs (as depicted in Fig. 2(a)), 4G eNBs, mobile edge computing (MEC) gateway, and edge computing servers, which were all deployed by AT&T. The 5G gNBs operate at 39 GHz, while the 4G eNBs operate at 700 MHz and 1.9 GHz. The architecture employs 3GPP NSA option 3 (as specified in Section II-A), where UEs are anchored to the operational core network over the existing LTE/EPC control plane.

The testbed implements a local breakout to the edge server. The on-site MEC gateway is installed between the cellular BSs (base stations) and AT&T's centralized LTE core. This network deployment option allows us to experiment with a 5G edge computing scenario since the local aggregation point directs

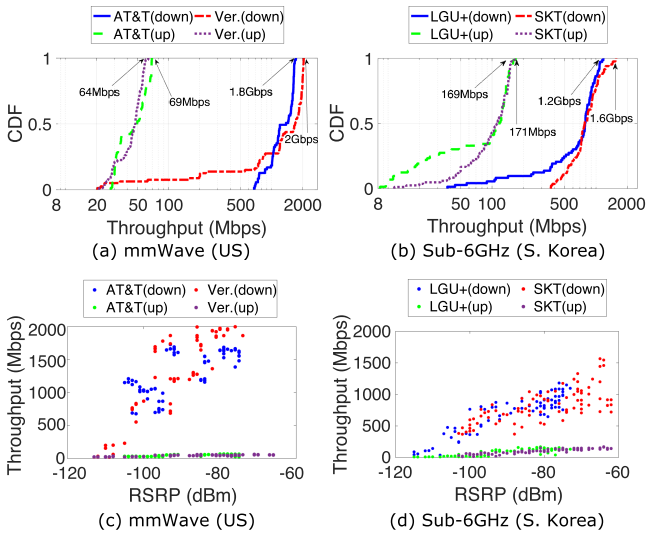


Fig. 4. Throughput measurements in 5G networks.

traffic to the edge computing servers placed right beside the 5G gNB. We utilize this testbed for the evaluation of transport protocols (Section V) and mobile applications (Section VII) in the mmWave-based 5G edge computing scenario.

IV. Characterizing 5G Networks

In this section, we characterize 5G performance in terms of throughput and latency. Our comprehensive measurements have been carried out across different frequencies (mmWave and sub-6 GHz), both directions (uplink and downlink), different operators (two from the U.S. and two from South Korea), and different server locations (operator's edge in the cellular core and 3 rd party cloud out of cellular network). Our observations are used as a basis for analyzing the performance achieved by transport protocols and applications (Sections V and VII).

A. Throughput Performance: Asymmetry Between Uplink and Downlink in 5G

We measure maximum throughput by using Speedtest [35] on operational 5G networks at multiple locations. We obtain numerous samples over a wide range of RSRP (reference signal received power).

mmWave 5G: Fig. 4(a) shows the CDF of throughput measurements on mmWave 5G networks of AT&T and Verizon. First, we find that there is a considerable gap between downlink (DL) and uplink (UL) throughput. In particular, the downlink capacity can reach up to 1.8 Gbps and 2.0 Gbps for AT&T and Verizon, respectively, while the uplink capacity can reach up to 69.2 Mbps and 63.8 Mbps, respectively. Interestingly, such a performance gap comes from the fact that (i) cellular operators use extreme TDD (time division duplex) slot ratio between DL and UL for mmWave 5G NR to maximize downlink speed[5] (e.g., 10:1) and (ii) DL is assigned a much wider bandwidth than UL considering the device capability (e.g., 400 MHz versus 100 MHz). Second, we see that they reveal a strong correlation between RSRP and capacity: the better RSRP, the higher throughput, which is shown in the two sub-figures (a and b) in Fig. 4. For example, with < -110 dBm, the downlink throughput can go down to 13.2 Mbps in Verizon's 5G. However, as we use the indoor installment of 5G gNB of AT&T, we measure the throughput over RSRP values of $[-105, -85]$dBm,[6] thus showing a relatively narrow throughput gap.

Sub-6 GHz 5G: Fig. 4(d) shows throughput distributions achieved by sub-6 GHz 5G networks, which is known to provide broad coverage compared to mmWave-based small cells [45]. Similar to mmWave 5G networks, we find a large gap between downlink and uplink throughput in sub-6 GHz 5G networks. For example, in SKT's 5G, the achieved downlink throughput ranges between 1.6 Gbps and 369 Mbps, while the uplink throughput varies between 171 Mbps and 11.5 Mbps. When the RSRP is smaller than -104 dBm, we observe the switching to 4G LTE cell on the 5G phone. Further, we see a wide variation in throughput at similar RSRPs due to background traffic from actual users in the sub-6 GHz cell.

TABLE II
Ping RTT Comparison (In *ms*) Over 5G/4G Networks, Which are Collected At Several Locations With Various RSRPs Ranging Between -60 dBm and -110 dBm

	AT&T@mmWave		Verizon@mmWave	SKT@3.6GHz
	Edge server	Cloud server	Cloud server	Cloud server
5G	6.1 ($\pm$ 2.5)	40.7 ($\pm$ 7.4)	31.6 ($\pm$ 8.6)	22.3 ($\pm$ 2.8)
4G LTE-A	17.2 ($\pm$ 1.9)	47.2 ($\pm$ 0.6)	42.3 ($\pm$ 2.8)	24.3 ($\pm$1.4)

B. Latency Performance: More Reduction in mmWave 5G Than Sub-6GHz 5G

We ran ping tests from a 5G phone to measure the base RTT without any cross traffic. Each measurement is repeated 50 times over different RSRP values with the same BS. Table II shows our measurement results over three 5G/4G LTE-A networks of AT&T, Verizon, and SKT. The two operators from South Korea have similar average throughput as shown in Fig. 4(d), we decide to use only one carrier for comparing performance of cloud server with mmWave 5G network in the US. In each case, we use a single nearby server to emphasize differences between 4G/5G and frequencies.

mmWave 5G: We observe that 5G reduces end-to-end latency over 4G despite the NSA core architecture. In fact, 5G significantly reduces the air latency, which can be deduced by comparing the RTT measurements between 5G and 4G LTE links to the edge server, which is shown in the AT&T's result of Table II. A main reason is that in the 5G NR standard [46], mmWave 5G uses a much wider bandwidth and a higher OFDM (orthogonal frequency division multiplexing) subcarrier spacing compared to 4G LTE system (e.g., 120 kHz versus 15 kHz), which enables short frame duration (0.125 ms versus 1 ms) and low latency. Furthermore, in Verizon's network, we analyzed per-hop latency to the Internet server via `traceroute` from the phone and find that the 1st-hop RTT shows a big gap between 5G and 4G LTE (9.1 ms versus 25.8 ms), which presumably contains not only RAN between the UE and BS but also core backhaul path. As Verizon's 5G uses the NSA architecture [47],

[5]We confirmed this with a few cellular providers.

[6]Due to transmit power regulations, the tested 5G phone cannot receive very strong signal indoors.

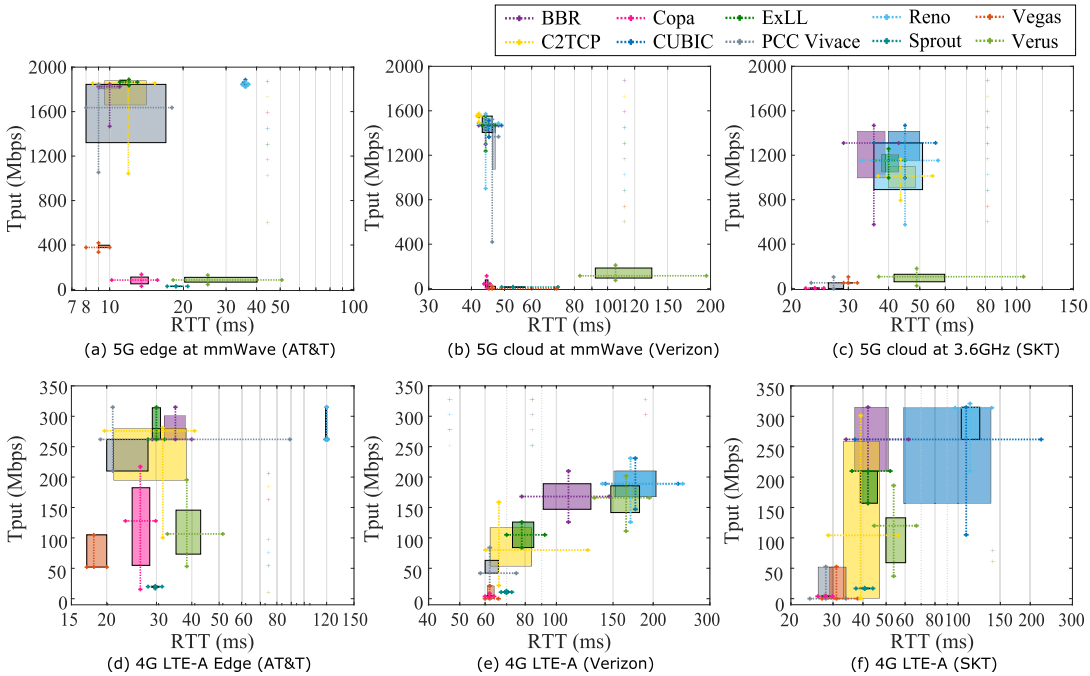


Fig. 5. Transport-layer CCA comparison in 5G/4G (writing consistency): Throughput versus RTT performance achieved by 10 CCAs under 5G networks in the U.S. (AT&T and Verizon) and South Korea (SKT). The right and bottom edges of the box represent the 25th percentile of the achieved RTT and throughput, respectively. The left and top edges give the 75th percentile. The two ends of the error bar give the 5th and 95th percentiles. The intersection point of the horizontal and vertical error bar represents the average of achieved RTT and throughput.

the primary reduction in latency stems from the mmWave-based 5G RAN [48].

Sub-6 GHz 5G: Interestingly, compared to mmWave 5G in the U.S., we do not see a significant gap in the latency between (sub-6 GHz) 5G and 4G networks deployed by SKT. We find the main reason behind such a small gain from (i) a similar OFDM subcarrier spacing and (ii) an edge-based cellular core architecture shared by 5G and 4G RANs [49].

V. Transport Protocol Performance

We evaluate the performance of transport-layer protocols in various practical scenarios using operational 5G networks. From the evaluation, we observe that the choice of a CCA can have a significant impact on the performance of 5G networks.

A. Performance With Edge Server: TCP Can Achieve Maximum Throughput

We first evaluate the effect of an edge server installed by an operator within its 5G/4G LTE-A networks. Specifically, with the mmWave 5G, the maximum throughput is 1.8 Gbps and the base RTT is 6 ms. On the other hand, with the 4G LTE-A, the maximum throughput is 276 Mbps and the base RTT is 17 ms.

Fig. 5(a) and (d) depict the performance comparison of ten CCAs obtained from the mmWave 5G and 4G LTE-A networks, respectively. As shown, we observe that TCP performance heavily depends on the choice of CCA in the cellular edge scenario. More importantly, we empirically confirm that recently developed CCAs can achieve near-optimal performance of both multi-gigabit throughput and sub-10 ms RTT without any parameter tuning or protocol modification. We analyze our results in detail below.

First, under near-zero loss environments, `CUBIC` achieves maximum throughput, but it results in an inflated RTT. Specifically, in 5G networks, `CUBIC` achieves 6 times the minimum RTT, while in 4G LTE-A (with carrier aggregation), it achieves 6.92 times the minimum RTT due to its aggressive buffer-filling mechanism until packet losses occur. Another loss-based window control CCA, `Reno`, also increases CWND until packet loss occurs. Second, despite achieving the minimum RTT, `Vegas` exhibits poor throughput performance: with a reduction of 22% and 33% from the maximum throughput over 5G and 4G, respectively. We find that it aims to maintain small CWND values for low RTT regime during the measurement. Third, `BBR` delivers high throughput and small RTT performance for both networks. In the mmWave 5G edge scenario, `BBR` shows throughput comparable to the loss-based CCA while achieving the average RTT within $2\times$. Surprisingly, we notice periodic peaks in RTT up to $10\times$ every 10 seconds when `BBR` returns to the previous state from the *ProbeRTT* phase, where it drops its rate to very low to drain the queue for at least 200 ms. This results in an 8.5% reduction in throughput from the maximum with substantial variations in both throughput and delay. Fourth, `ExLL` provides low-latency performance by maintaining the RTT within $2\times$ while achieving near-maximum throughput. It can calculate the maximum throughput estimate within 10 ms interval ($\approx$1RTT).

Fifth, `C2TCP` achieves similar throughput with a 10.5% deduction from `CUBIC` and maintains RTT within $1.7\times$ due to its delay-based throttling, which reduces CWND to a small value whenever the measured RTT exceeds $2\times$. As a result, it reveals some variations in both throughput and RTT. Sixth, `Verus` achieves 90.2 Mbps and 35.2 ms on average because it reveals bursty packet injection patterns in high-speed networks due to its fixed injection interval of 5 ms and allows up to $6\times$ RTT inflation by default. Fig. 6(a) illustrates the sample traces of CWND, RTT, and throughput achieved by the five CCAs over

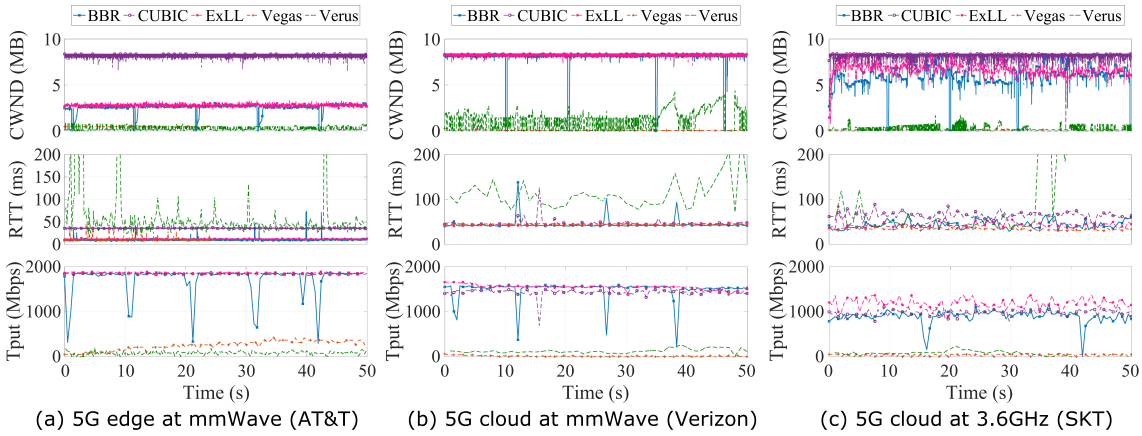


Fig. 6. CWND, RTT, and throughput traces over time achieved by five CCAs in 5G. Here, the CWND is bounded by the advertised RWND.

TABLE III
5G/4G LTE FREQUENCY BANDS USED AT TESTING LOCATIONS IN THE U.S. (AT&T AND VERIZON) AND SOUTH KOREA (LGU+ AND SKT)

	AT&T [50]	Verizon [43]	LGU+ [51]	SKT [44]
5G	39GHz	28/39GHz	3.4GHz	3.6GHz
4G LTE	700MHz/1.9GHz	700MHz/1.9GHz	800MHz/2.6GHz	800MHz/2.6GHz

time and corroborates our analysis. Lastly, the other four CCAs, including `Copa`, `PCC Vivace`, and `Sprout`, underperform compared to the rest, so we exclude them from our performance evaluation in Section V. We discuss this in Section VIII.

B. Performance With Cloud Server: TCP Cannot Achieve Maximum Throughput

For evaluation with an Internet cloud server, we use two different cellular networks deployed by Verizon and SKT, whose frequency bands are shown in Table III. In particular, with Verizon's mmWave 5G, the maximum throughput is 2 Gbps and the minimum RTT is 32 ms (thus, BDP is 8 MB), and with the 4G LTE-A, the maximum throughput is 189 Mbps and the minimum RTT is 43 ms (BDP is about 1 MB). Also, with SKT's sub-6 GHz 5G, the maximum throughput is 1.6 Gbps and the minimum RTT is 22 ms (i.e., BDP of 4.4 MB), and with the 4G LTE-A, the maximum throughput is 314 Mbps and the minimum RTT is 24 ms (i.e., BDP of 942 KB).

Fig. 5(b) and (e) present the performance results achieved by ten CCAs from the cloud server to 5G phone over Verizon's mmWave 5G and 4G LTE-A networks. In the mmWave 5G case, we observe a very interesting point. Although the maximum achievable throughput is 2 Gbps, the top three CCAs (`BBR`, `CUBIC`, and `ExLL`) can only achieve around 1.5 Gbps along with an average RTT less than $1.5\times$. Surprisingly, the main reason for such under-utilization is that the maximum RWND[7] at the receiver (i.e., 5G smartphones) limits the CWND of the TCP sender, as shown in Fig. 6(b). This originates from the Android phone's TCP setting: `tcp_rmem_max` is set to 8 MB and 4 MB for 5G and 4G, respectively. Furthermore, we find that the RTT performance is effectively controlled by the maximum window size via TCP's flow control mechanism. However, this ad-hoc setting cannot work well in the case of 4G LTE-A where the configured value is much larger than the BDP (4 MB versus 1 MB). As a result, we see different performance points between the three algorithms: `BBR` and `ExLL` provide an excellent tradeoff between throughput and RTT compared to `CUBIC` (Fig. 5(e)).

Also, we find that a delay-based CCA, `Vegas` performs worse than in the 5G edge setting. `Vegas` achieves the smallest throughput at around 10 Mbps due to larger queueing delay compared to the observed RTT, thus limiting the CWND increase under no packet loss as studied in [53]. `Verus` operates in the region of low throughput of less than 200 Mbps on average in the path with typical RTT values (25∼45 ms) since it can adjust the number of packets to send using a fixed 5 ms window over those RTTs.

On the other hand, Fig. 5(c) and (f) compare the performance of ten CCAs over SKT's sub-6 GHz 5G and 4G LTE-A networks, respectively. Similarly to the mmWave 5G cloud scenario, we find that a loss-based CCA, `CUBIC`), attains the highest throughput but at the cost of increased RTT of up to $2.1\times$ due to a certain queue buildup (but limited by `tcp_rmem_max`). Overall, we confirm the effectiveness of RWND setting at the client-side (i.e., 5G phones) against bufferbloat [54], [55]. In addition, we observe severe performance variations compared to the previous measurements obtained in the U.S. One primary reason is that there exist much more active users sharing the same frequency resources in the 5G/4G networks, which leads to more delay variations in multi-user scheduling, despite smaller base RTTs. This is especially true considering the wide coverage of sub-6 GHz 5G and the high number of mobile subscribers in South Korea.

Remark on AT&T's 5G. We evaluated different CCAs with a CloudLab server on the Internet in AT&T's 5G network. However, we observed that the maximum throughput achieved by a single flow was throttled by 1 Gbps.[8] Consequently, most high-performance algorithms showed similar results close to the

[7]Current Linux TCP implementation adopts a receive buffer auto-tuning technique called dynamic right sizing (DRS) [52], so the RWND increases twice over time up to `tcp_rmem_max`; DRS increases the RWND size only when it might potentially limit the CWND growth but never decreases it.

[8]We believe that the operator might throttle the throughput per flow with around 1 Gbps at the EPC side, as we could achieve the maximum throughput of 1.8 Gbps using multiple flows.

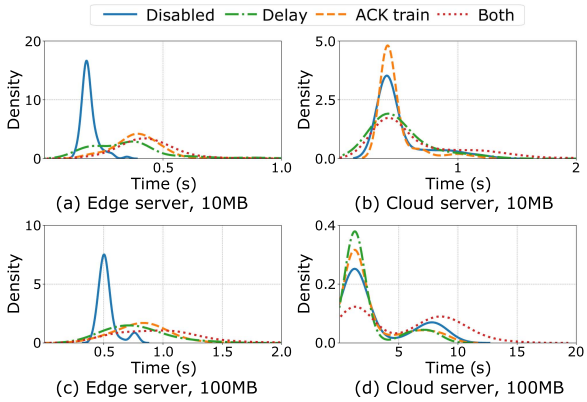


Fig. 7. File download times with CUBIC's HyStart algorithm. Disabled means it uses the standard Slow Start algorithm. Hystart uses delay changes, ack train length, or both.

achievable maximum throughput and the minimum RTT since their packets in flight were limited by a fixed value lower than BDP. However, we omit the details due to space constraints.

C. Slow Start Performance

TCP's slow start phase exponentially increases the congestion window for every round-trip time (RTT) until it reaches the slow-start threshold (`ssthresh`). However, CUBIC uses a modified slow-start mechanism called Hybrid Slow Start (`HyStart`) to prevent excessive packet loss. Ha et al. [56] proposed using two indicators, the ACK train length, and the increase of RTT delays, to determine when to exit the slow-start phase and switch to the congestion avoidance phase. As CUBIC is used as the default CCA in Linux servers, we evaluated the performance of its slow start algorithm.

To evaluate the performance of different `HyStart` modes, we conducted experiments with four different settings as shown in Fig. 7: Disabling `HyStart` and enabling `HyStart` with delay threshold, ACK train, or both. Our experiments involved downloading files of various sizes, ranging from 1 KB to 100 MB, using the `Nginx` web server and `Curl`. We measured the transfer times of each file 100 times.

Our results show that disabling `HyStart` results in aggressive CWND growth, leading to the fastest downloading speed when a user downloads files from the edge server, but at the cost of higher packet loss rates (Fig. 7(a) and (c)). Even with high loss rates, TCP can recover from them relatively quickly, thanks to a relatively small number of packets in flight with lower RTTs. Enabling `HyStart` with either ACK train or delay threshold improves the overall downloading times compared to the standard slow start, as shown in Fig. 7(b) and (d), while enabling `HyStart` with both ACK train and delay threshold shows the worst download times, which requires further investigation.

VI. Performance Under Dynamics: TCP Cannot Guarantee Low Latency

As the primary deployment in commercial 5G networks has focused on providing outdoor coverage in densely populated areas (e.g., Verizon [43]), we aim to explore the behavior of different CCAs in the presence of channel dynamics. For realistic evaluations, we focus on two case studies: (i) transient channel degradation due to temporary obstacles that block the line of sight (LOS) channel between the 5G cell tower and UE, and (ii) continuous channel fluctuation due to user mobility and handover between 5G and 4G cells. We believe that our case studies cover many feasible situations that users may encounter, especially in the dense urban deployment of 5G NRs [57], [58].

A. Throughput Fluctuation for Stationary UEs

We investigate how each CCA reacts to practical obstructions that cause NLOS (Non-Line-of-Sight) channel conditions in the 5G link. As a baseline, we position the UE 50 m away from the 5G cell tower and establish a TCP connection in the downlink using iperf3. To create obstructions, we use two representative obstacles: the human body and a vehicle. For the former, while downloading data, a human walks through the LOS path and pauses for 10 seconds to obstruct the signal. We repeat this process two times at 20 and 40 seconds during the experiment. For the latter, we move the phone behind a vehicle for 10 seconds and then return it to its original position. We also repeat this process two times at 20 and 40 seconds, as before. We conduct this experiment in the mmWave 5G network as we could not observe any noticeable impact from those obstacles in the sub-6 GHz 5G link due to better radio propagation characteristics [59].

Blockage by Human Body: We found that the presence of even a single human body can cause a significant degradation in mmWave channel quality by blocking the LOS channel, leading to an average drop of 26 dB in RSRP. This significant reduction in channel quality results in a considerable decrease in throughput of up to 50% (from 2 Gbps to 1 Gbps) in all CCAs we tested except `Verus`, as depicted in Fig. 8(a). However, we observed significant differences in RTT among the CCAs. Loss-based schemes showed the highest increase in RTT, up to 5× from 21 ms to 105 ms. While `BBR` showed an RTT increase of 2× during the blockages, `ExLL` was able to maintain the lowest RTT (1.3×) regardless of the blockages, outperforming all other CCAs in latency. We did not observe any degradation due to blockage in `Verus`, likely because it was under-utilized.

Blockage by Vehicle: We observed a more severe impact on the 5G link when the UE was obstructed by a vehicle, which causes a deep fading of about 29 dB in RSRP due to the vehicle's thick metallic body. All CCAs, except for `Verus`, suffered from significant throughput degradation. As shown in Fig. 8(b), the throughput dropped from 2 Gbps to around 350 Mbps on average. Furthermore, we observed similar patterns in the RTT performance as in the case of obstruction by a human body. Loss-based CCAs experienced the largest increase in RTT, up to 5× from 21 ms to 105 ms. `BBR` showed an RTT increase of 2× during the blockages. In contrast, `ExLL` maintained the lowest RTT, only increasing by 1.3× on average, outperforming all other CCAs in terms of latency.

Loss Due to Blockage: When there is a complete blockage in the LOS channel, heavy packet losses can occur, leading to a catastrophic throughput loss that approaches zero. This situation is shown in Fig. 8(c). TCP responds to this by drastically reducing its CWND to a very small value (up to 140× reduction). Therefore, it is crucial for a CCA to recover its original performance quickly once the blockage disappears. To evaluate

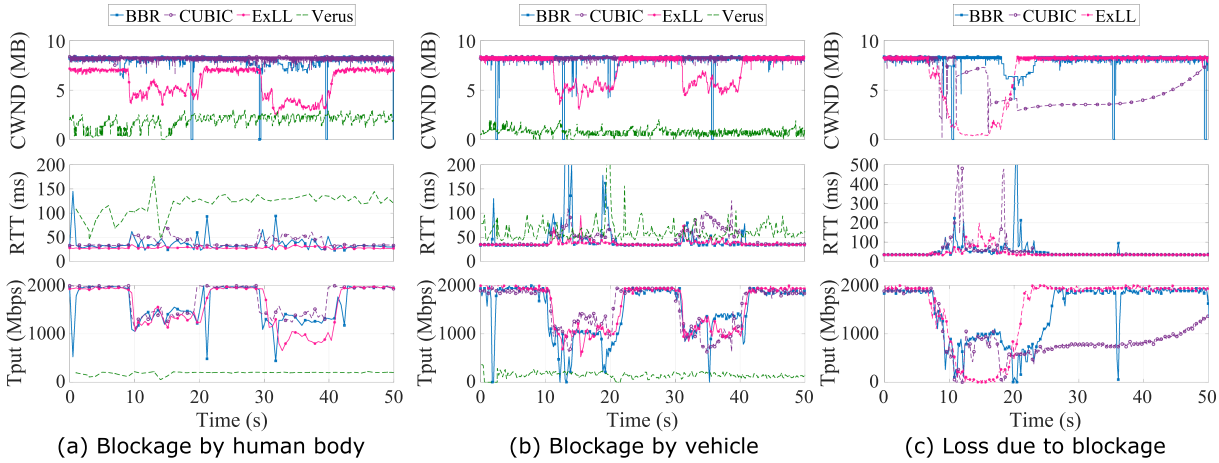


Fig. 8. Blockage scenario in Verizon's mmWave 5G network: Throughput trace of different CCAs over time.

TABLE IV
COMPARISON OF RECOVERY TIME FOR DIFFERENT CCAS FROM 10 S BLOCKAGE

	CUBIC	BBR	ExLL	Reno
Recovery time (sec)	30.6	5.4	3.1	184

each CCA's recovery time, we measure the time it takes for them to fully restore their achievable capacity after experiencing complete obstruction. Table IV compares the recovery time for three schemes, along with `Reno`'s results as a reference [60]. The best performing CCA (`ExLL`) takes 3.1 seconds to fully restore its reduced throughput. In contrast, the others take at least 1.7× longer time, indicating that new CCA designs that focus on rapid adaptation to such dramatic variations in 5G or beyond are necessary.

B. Throughput Fluctuation for Mobile UEs

Another source of cellular link capacity variations arises from channel quality variations caused by user mobility. To this end, we test two different mobility cases in commercial 5G networks: a driving scenario in the U.S. and a subway scenario in South Korea. We used Samsung Galaxy S20 Plus/Ultra (Qualcomm Snapdragon 865 [37]) for a driving scenario in the U.S.'s mmWave 5G networks and Galaxy S10 (Snapdragon 855 [38]), and Galaxy Note 10 Plus is used for a subway scenario in South Korea's 5G networks at sub-6 GHz bands.

Driving Scenario: We put the phone in a car at a downtown location with RSRP of −85 dBm, and after 30 seconds, we drive along a predefined trajectory to another location with RSRP of −105 dBm in the next 2.5 minutes. We then drive back to the starting location (−85 dBm in RSRP), taking about 2.5 minutes, and remaining stationary for 30 seconds. In total, a single driving test takes 6 minutes. We repeat the same process five times for each CCA. In this driving test, we observe a handover event every 33.3 seconds on average and RSRP fluctuation ranging between −64 ∼ −109 dBm. The gNBs in the route are 250 – 300 m apart from each other.

Fig. 9(a) shows the CDF of achieved throughput between four CCAs in vehicular mobility. We find that `BBR` and `ExLL` work well against severe channel variations over time. `BBR` performs the best due to its quick adaptation logic, irrespective of packet losses. `ExLL` react to RTT variations carefully as it injects the packets conservatively to maintain low latency. On the other hand, `CUBIC` cannot utilize channel resources in dynamic conditions because it keeps reducing CWND by consecutive packet losses during the handover between (small cell) mmWave gNBs and LTE eNBs. `Verus` shows a slow adaptation and low link utilization under the 5G link's high dynamics. Fig. 9(b) compares the CDF of RTT for those algorithms, where `ExLL` achieves the lowest RTT compared to the others.

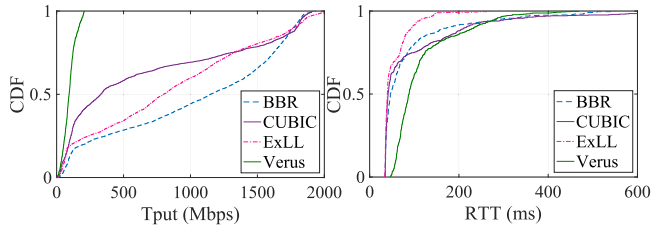


Fig. 9. Driving scenario over Verizon's 5G in U.S.

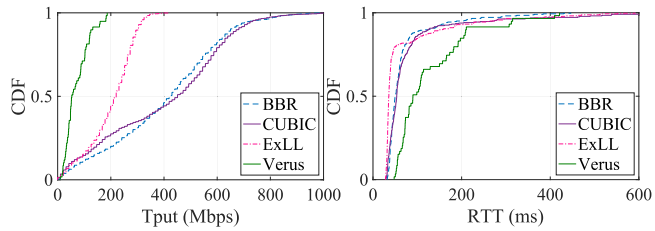


Fig. 10. Subway scenario over SKT's 5G in South Korea.

Subway Scenario: In South Korea, major operators have completed the establishment of 5G connectivity on subway lines of major cities recently [61]. We evaluate each CCA while the subway is moving on the same route. In this scenario, a handover event occurs every 14 seconds on average, and RSRP varies between −66 ∼ −98 dBm. We observe that the cellular connection temporarily switches to 4G LTE when crossing two 5G cells due to the NSA architecture of the current 5G NR deployment.

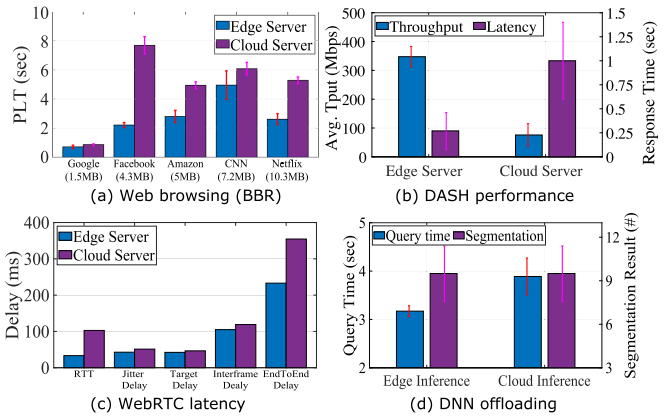


Fig. 11. The results of Application performance measurements.

Fig. 10 depicts the CDF of throughput and RTT performance for four CCAs. One noticeable difference from the driving test is that the loss-based CCAs perform well in throughput over the others. Specifically, CUBIC achieves the best throughput. We find that HARQ (hybrid ARQ) conceals packet losses successfully in most cases. Then it leads to the situation where CUBIC maintains large CWND values during the experiment. BBR attains comparable performance with CUBIC in both throughput and RTT. ExLL effectively suppresses RTT inflation but accompanies some throughput loss due to its conservative behavior under a highly varying RTT condition. Verus still exhibits low channel utilization with inflated RTT due to sender-side queueing from the bursty transmission pattern.

VII. Application Performance

In this section, we will evaluate the potential improvement of popular mobile applications when provided with edge services via our mmWave 5G network testbed (Section III-C). For comparison purposes, we will conduct the same test with the CloudLab server on the Internet. Based on our evaluation, we have observed two critical points. First, despite the substantial latency gain from the edge services, the performance of the application is still hindered by various processing tasks that run on the 5G phone. Second, the application that relies on uplink delivery is unable to leverage the latency gain fully due to the much lower bandwidth in uplink compared to downlink.

A. Web Browsing: Edge Reduces Page Loading Time

Setup: In this test, we select five popular websites from Alexa top sites: Amazon (5 MB), CNN (7.2 MB), Facebook (4.3 MB), Google (1.5 MB), and Netflix (10.3 MB). We crawl their landing pages and host them on our edge/cloud server. To conduct the experiment, we connect a 5G phone to a laptop and use the Chrome remote debugging framework [62] on the laptop to extract page loading information. We use HTTP/2.0 and NginX webserver. We load each website back-to-back using the 5G phone over 5G from both the edge server and cloud server. We clear the web cache for cold-cache loading of all sites and measure their page loading time (PLT). Each experiment is repeated five times, and the average is reported with standard deviation.

Results: Fig. 11(a) presents a comparison of the PLT of five web pages when downloading from the edge and CloudLab servers (6.1 ms versus 40.7 ms in ping RTT) using BBR. The results confirm a latency reduction ranging from 18.3% to 71.2%, depending on the web page's size, when we use the edge for hosting a web server. However, considering the RTT difference from UEs to edge or cloud servers, the improvement in application performance is lower than what we expect from the latency gain in the network. In web browsing, parsing and rendering processes can make the application-level performance less improved since they take as much time as downloading. To check the gap from different CCAs, we also tested CUBIC and ExLL, which show similar gains between 29.1% $\sim$ 73% and 19.5% $\sim$ 70.9%, respectively. Due to the relatively small size of web contents (e.g., $<$10 MB) compared to 5G's gigabit capacity, it is hard to see a significant gap between CCAs for the web browsing application.

According to a study by Li et al. BBR's probing phase was found to result in higher throughputs than CUBIC's slow-start mechanism [63]. However, we also observe that this gain is limited by the slow start phase of CCAs since the download of a few MB webpages finishes before the CWND reaches the 5G link's capacity.

B. HTTP Adaptive Streaming: Edge Reduces Video Response Time

HTTP-based adaptive streaming (standardized as DASH [64]) is one of the dominant applications accounting for mobile video traffic. High bandwidth in 5G will accelerate the use of high-resolution video streaming such as 4 K DASH streaming.

Setup: For DASH streaming, we use the "Big Buck Bunny" test video [65] encoded at bitrates in {4, 8, 12}Mbps, corresponding to {720, 1080, 2160}p (i.e., HD, FHD, 4 K, respectively) at 30 frames per second (FPS). Also, we create an MDP (media presentation description) file with a 4 s chunk duration for a total length of 120 seconds. In our setup, we use a Chrome browser that can directly run dash.js (v3.2.1) [64] to stream videos on the phone. We also set our DASH video server using Node.js that runs on either the edge server or CloudLab server. For comparison, we measure the average throughput at the player and calculate the response time from when the client makes a streaming request for each chunk until the chunk download completes for display on the client screen.

Results: Fig. 11(b) shows performance comparison in the case of 4 K video streaming. We find that DASH with an edge server can substantially reduce the response time by 72.1% on average compared to DASH with the CloudLab server. It is because the sub-10 ms short RTT ($\approx$6 ms) between the 5G phone and edge server quickly ramps up the actual throughput while downloading each 5 $\sim$ 7 MB chunk, leading to a 4.5$\times$ increase in the achieved throughput as shown in Fig. 11(b). Note that we can still see a significant gain for videos with lower resolution: the response time reduction is 78.6% and 79.9% with FHD and HD, respectively.

C. Live Video Telephony: Processing Task Can Be the Bottleneck

Delay is critical to live video telephony [66], [67]. In this context, we evaluate the potential gain of using edge computing for live video streaming applications.

Setup: We use WebRTC [68], an open-sourced de facto standard platform supported by major web browsers, to evaluate live video telephony. For a pair of clients, we use a high-performing laptop (CPU 12-core at 2.6 GHz, RAM 15 GB, Ubuntu 18.04) with a USB tethered 5G phone, since we want to avoid the impact of video processing bottleneck in the phone as observed in [8]. We modify the stand-alone WebRTC code to use our relay server at the edge. For comparison, we also install our relay server in the CloudLab server to measure the video streaming performance. WebRTC is configured to stream video from the camera with 1080P (FHD) resolution and 30FPS at 6 Mbps. For evaluation, we use five metrics: RTT at the transport layer, jitter buffer delay, target delay, inter-frame delay, and end-to-end streaming delay. We extract all metrics from our modified WebRTC.

Results: Fig. 11(c) shows the latency performance of WebRTC with FHD video via two relay servers at the edge inside the 5G core and the Internet cloud. We find a substantial gain of 67.7% in the actual RTT measured at the transport layer (33.2 ms versus 102.8 ms). However, in terms of the end-to-end latency, we see a benefit of 51.9% on average (233 ms versus 354.6 ms), compared to that in the transport layer. The gain in the streaming latency is due to a non-negligible amount of per-frame processing delay for video capture, encoding, decoding, and rendering tasks.

We expect that the latency reduction will diminish further due to increased processing complexity in higher-quality live streaming such as real-time 360° video [8], [67] and 4 K video [69]. Thus, our result encourages more research on the innovative video codec that can support high-resolution video over the 5G edge for ultra-low-latency streaming.

D. Deep Neural Network Offloading: 5G Uplink is the Bottleneck

Deep neural network (DNN) offloading is one promising area for cloud/edge computing, which allows the mobile client to make a large-scale DNN inference with the help of the edge [70], [71], [72].

Setup: We consider a semantic segmentation task for images containing multiple objects with a pyramid scene parsing network (PSPNet) [73]. We implement an Android application that can offload a DNN inference by uploading the image file to either the edge or cloud server, downloading the inference result from the server, and displaying it on the screen. The client and server communicate over HTTP. We implement the DNN inference using the Keras library in Python on the server and use the image dataset from Cityscapes [74]. In this setting, we measure the query execution time, the time from offloading the DNN inference to the edge server or the CloudLab, to display the result on the phone.

Results: Fig. 11(d) shows the query execution time and segmentation results when offloading 50 segmentation tasks on the edge and the CloudLab server, respectively. We observe a reduction of 18.5% on average in the latency while achieving the same inference accuracy. The server execution time of ≈3 ms is comparable despite different CPU specifications (i.e., 4 cores at 1.6 GHz edge server versus 8 cores at 2.4 GHz CloudLab server). The performance gain comes from the decreased latency on the end-to-end path between the 5G phone and the edge server. However, the gain is severely limited due to the low uplink throughput compared to the high downlink throughput, which stems from the operator's deliberate setting (Section IV).

TABLE V
PERFORMANCE COMPARISON IN FILE UPLOAD

Upload		200KB	500KB	1MB	2MB	5MB
Edge	Transfer time (sec)	0.03	0.04	0.21	0.56	1.5
	Effective tput (Mbps)	53.3	100	38.1	28.6	26.7
Cloud	Transfer time (sec)	0.1	0.12	0.37	0.75	1.68
	Effective tput (Mbps)	16	33.3	21.6	21.3	23.8

TABLE VI
PERFORMANCE COMPARISON IN FILE DOWNLOAD

Download		200KB	500KB	1MB	2MB	5MB
Edge	Transfer time (sec)	0.12	0.15	0.21	0.23	0.32
	Effective tput (Mbps)	13.3	26.7	38.1	69.6	125
Cloud	Transfer time (sec)	0.28	0.36	0.39	0.51	0.62
	Effective tput (Mbps)	5.7	11.1	20.5	31.4	64.5

E. File Transfer: Large Gain in Downlink versus Small Gain in Uplink

Setup: We evaluate the transfer time for uploading and downloading files with different sizes ranging from 200 KB to 5MB[9] using iperf3 and compare the performance when using the edge and cloud server, respectively. We measure the average transfer time over five trials, and the effective throughput is calculated as the data size divided by the transfer time.

Results: Tables V and VI present the measurement results and we make two important observations. First, the 5G link exhibits very distinct patterns between uplink and downlink due to an order-of-magnitude gap, as observed in Fig. 4. Specifically, the BDP of 5G uplink is about 52 KB, which is much smaller than 1.35 MB of 5G downlink (i.e., 26× gap). Therefore, the effective throughput using 5G uplink starts decreasing with the increasing data beyond 500 KB, while the throughput over 5G downlink keeps increasing with the increasing data size. Second, as the data size increases, the benefits of using the edge become more pronounced, but only in the downlink. Consequently, emerging applications with heavy uplink traffic would not be able to fully leverage the benefits of the 5G edge service. Note that this issue manifests itself in 5G over 4G.

VIII. DISCUSSION

Applicability to the 5G Standalone (SA) Deployment: As mentioned in Section II-A, most operators currently reuse the existing LTE core for 5G services with the NSA option and are expected to deploy 5G SA for private enterprise networks in the near future [76], [77]. One of the major use cases can be ultra-reliability and low latency services (URLLC) over such networks. As our mmWave-based 5G testbed along with AT&T's edge computing server is characterized by >1 Gbps bandwidth and sub-10 ms latency in the path between two endpoints, we believe that our results can offer insight as to how each CCA would operate in the pure 5G with SA architecture.

Overheating Issues: We observe that 5G phones fail to maintain a persistent connection with 5G gNB even when the phone

[9]This is a typical data size for mobile cloud storage services [75].

is stationary with stable RSRP values due to overheating in the mmWave RF module [78]. These modules overheat quickly at Gbps speeds, which leads to the cell switching to 4G LTE and abruptly and substantially degrading the throughput. We observed this cell switching whenever the 5G RF module's temperature exceeded 65 °C. We measured the phone's internal temperature by reading the thermal _ zone value.[10] It takes a long time (more than 60 seconds) to dissipate the heat, and thus heat management solutions need to be devised for 5G phones with mmWave RF modules, as suggested in [79].

IX. Related Work

5G Measurements: As 5G deployment is still in an early stage, only a few studies have been done. Narayanan et al. [7] conducted extensive field tests on commercial 5G performance in three U.S. cities as a baseline for further research. They found that severe throughput fluctuation in 5G can happen due to obstruction, weather, and frequent handover, compared to 4G LTE. Since then, they proposed Lumos5G, a context-aware machine learning framework that predicts 5G throughput [58]. They illustrated the performance, power, and QoE implications [9]. Moreover, they measured the beam management and signal propagation characteristics of commercially deployed mmWave 5G networks using a professional tool named Accuver XCAL from a recent paper [80]. Meanwhile, Xu et al. [8] demystified 5G networks at 3.5 GHz through cross-layer measurements by analyzing several perspectives from physical layer characteristics to application performance, including energy consumption. Their preliminary TCP experiment showed that legacy TCP could lead to low utilization due to excessive packet drops. Recently, Hassan et al. [81] compared the handover mechanisms between 4G and 5G and proposed a handover prediction system.

Measurement on Edge: Edge computing in 5G network is one of the hot topics which has been recently investigated. The most recent work done by the authors in [10] presented a measurement of the performance of public edge platforms using various kinds of wireless networks such as wifi, LTE, and 5G. They compared the commodity edge platforms with the cloud platforms, in terms of the end-to-end network delay, throughput, and the application QoE. However, they have not evaluated an in-depth comparison between different TCP CCAs in a full-fledged 5G edge setting.

Performance Study of CCAs on 5G Networks: Zhang et al. [6] have shown that the throughput performance of edge servers using four different CCAs is slightly better than that of remote servers. However, the latency performance is dramatically improved on the edge servers. Furthermore, our study showed that BBR and CUBIC had the highest throughput, consistent with previous evaluations of these algorithms in various network conditions. BBR also achieved lower latency than CUBIC, consistent with studies comparing the latency performance of different TCP congestion control algorithms.. Several simulation-based studies have also investigated TCP congestion control in 5G networks. Ford et al. [82] identified several design issues regarding core architecture, wireless scheduling, and congestion control along with a quantitative analysis using an ns-3 simulator for mmWave 5G networks. Zhang et al. [6] presented a comprehensive simulation study of TCP in mmWave 5G networks, including four CCAs, edge versus remote servers, and handover. They showed that TCP performance on mmWave links is highly dependent on different combinations of these settings. Ichkov et al. [83] analyzed end-to-end mobile UE performance in urban mmWave 5G networks in the presence of dynamic pedestrian blockages, which shows that TCP throughput is severely affected by frequent short-term blockages in the mmWave link. Haile et al. [84] conducted research comparing CCAs over QUIC with 5G-trace emulations and live network experiments in a 5G testbed. They measured throughput and RTT of RBBR [85], BBR, Copa, and CUBIC. Copa maintains low latency but underperforms in throughput compared to other CCAs. CUBIC achieved comparable throughput to BBR, but the RTT was too high. Overall, their results were very similar to ours. However, the evaluation's scope is restricted by the 1 Gbps maximum bandwidth.

[10] adb shell cat sys/class/thermal/thermal _ zone∗/temp

X. Conclusion

We have performed the first in-depth measurement study using the edge computing server over operational 5G networks. We have comprehensively evaluated ten different CCAs using off-the-shelf 5G phones in terms of throughput and latency performance. We found that each CCA has its pros and cons depending on the 5G network characteristics and 5G phone configurations, which calls for a better congestion control algorithm tailored for 5G/6G cellular networks. We also assessed the efficacy of the edge server for various mobile applications. We believe that our measurement-driven findings can encourage further research efforts to use commercialized 5G infrastructures effectively. In future work, we plan to explore ways to improve congestion control algorithms to maximize throughput and minimize latency in 5G infrastructure and edge computing.

References

[1] GSA, "LTE & 5G Market Statistics – March 2021," 2021. [Online]. Available: https://gsacom.com/paper/lte-5g-market-statistics-march-2021/

[2] 3GPP, "Release 15," Apr. 2019. [Online]. Available: https://www.3gpp.org/release-15

[3] J. Huang et al., "An in-depth study of LTE: Effect of network protocol and application behavior on performance," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 363–374.

[4] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, "Comparison of caching strategies in modern cellular backhaul networks," in *Proc. ACM 11th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2013, pp. 319–332.

[5] M. Polese, R. Jana, and M. Zorzi, "TCP and MP-TCP in 5G mmWave networks," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 12–19, Sep. 2017.

[6] M. Zhang et al., "Will TCP work in mmWave 5G cellular networks?," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 65–71, Jan. 2019.

[7] A. Narayanan et al., "A first look at commercial 5G performance on smartphones," in *Proc. Web Conf.*, 2020, pp. 894–905.

[8] D. Xu et al., "Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption," in *Proc. Annu. Conf. ACM Special Int. Group Data Commun. Appl. Technol. Architectures Protoc. Comput. Commun.*, 2020, pp. 479–494.

[9] A. Narayanan et al., "A variegated look at 5G in the wild: Performance, power, and QoE implications," in *Proc. ACM SIGCOMM Conf.*, 2021, pp. 610–625.

[10] M. Xu et al., "From cloud to edge: A first look at public edge platforms," in *Proc. 21st ACM Internet Meas. Conf.*, 2021, pp. 37–53.

[11] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, and K. Lee, "ExLL: An extremely low-latency congestion control for mobile cellular networks," in *Proc. ACM 14th Int. Conf. Emerg. Netw. EXperiments Technol.*, 2018, pp. 307–319.

[12] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[13] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, pp. 64–74, Jul. 2008.

[14] 3GPP, "5G; NR; Base Station (BS) radio transmission and reception (3GPP TS 38.104 version 16.4.0 Release 16)," 2020. [Online]. Available: http://www.3gpp.org/dynareport/38104.htm

[15] GSMA, "5G implementation guidelines: NSA option 3," Feb. 2020. [Online]. Available: https://tinyurl.com/bdh6ksj2

[16] O. N. C. Yilmaz, O. Teyeb, and A. Orsino, "Overview of LTE-NR dual connectivity," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 138–144, Jun. 2019.

[17] 3GPP, "Universal Mobile Telecommunications System (UMTS); LTE; 5G; NR; Multi-connectivity; Overall description; Stage-2 (3GPP TS 37.340 version 15.9.0 Release 15)," 2020. [Online]. Available: http://www.3gpp.org/dynareport/37340.htm

[18] The Week Magazine, "South Korea launches world's first commercial 5G service," 2019. [Online]. Available: https://tinyurl.com/382z6k4r

[19] Forbes, "Verizon launches world's first commercial 5G smartphone service," 2019. [Online]. Available: https://tinyurl.com/5dkh4hh7

[20] GSMA, "5G - Future networks," Sep. 2020. [Online]. Available: https://tinyurl.com/4cp63meb

[21] Beebom, "List of 5G bands in the US for verizon, AT&T, sprint and T-mobile," Jan. 2021. [Online]. Available: https://beebom.com/list-5g-bands-us-verizon-att-sprint-t-mobile/

[22] R. News, "Is 5G a gamechanger for end-user experience?," Mar. 2020. [Online]. Available: https://tinyurl.com/y23oee74

[23] RCRWireless News, "Korean telcos aim to launch mmwave 5G network in 2020: Report," 2020. [Online]. Available: https://tinyurl.com/yxqtkhzm

[24] Opensignal, "SOUTH KOREA 5G user experience report," Jun. 2020. [Online]. Available: https://tinyurl.com/u3rxvzbh

[25] J. Gettys, "Bufferbloat: Dark buffers in the Internet," *IEEE Internet Comput.*, vol. 15, no. 3, pp. 96–96, May/Jun. 2011.

[26] P. Balasubramanian, "Updates on windows TCP," 2017. [Online]. Available: https://tinyurl.com/ex627de9

[27] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the Internet," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, 2018, pp. 329–342.

[28] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 459–472.

[29] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. ACM Conf. Commun. Architectures Protoc. Appl.*, 1994, pp. 24–35.

[30] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proc. ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 509–522.

[31] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1246–1259, Dec. 2006.

[32] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proc. IEEE Conf. Comput. Commun.*, 2000, pp. 1157–1165.

[33] 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, Apr. 2019.

[34] D. Duplyakin et al., "The design and operation of CloudLab," in *Proc. USENIX Conf. Usenix Annu. Tech. Conf.*, 2019, pp. 1–14.

[35] Ookla, "SPEEDTEST," 2020. [Online]. Available: https://www.speedtest.net

[36] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu, "iPerf3: A TCP, UDP, and SCTP network bandwidth measurement tool," 2019. [Online]. Available: https://iperf.fr

[37] Qualcomm, "Snapdragon 865 5G mobile platform," Dec. 2019. [Online]. Available: https://tinyurl.com/mv455umf

[38] Qualcomm, "Snapdragon 855 5G mobile platform," Dec. 2018. [Online]. Available: https://tinyurl.com/2bmbhp6p

[39] Samsung, "Exynos 9825," Aug. 2019. [Online]. Available: https://tinyurl.com/yckz5fea

[40] Google, "Android NDK r21d," Jun. 2020. [Online]. Available: https://developer.android.com/ndk

[41] T. L. Foundation, "IP-sysctl.txt in kernel docs - The Linux kernel archives," Jul. 2020. [Online]. Available: https://tinyurl.com/bdd67fv3

[42] Syslabshare, "5G measurement tools," 2021. [Online]. Available: https://github.com/syslabshare1/5G_measurement_tool

[43] Verizon, "5G coverage map," Jan. 2021. [Online]. Available: https://www.verizon.com/5g/coverage-map/

[44] SKT, "5G coverage map," Jan. 2021. [Online]. Available: https://tinyurl.com/y4n9gc53

[45] E. Bjornson, L. Van der Perre, S. Buzzi, and E. G. Larsson, "Massive MIMO in Sub-6GHz and mmWave: Physical, practical, and use-case differences," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 100–108, Apr. 2019.

[46] ShareTechnote, "5G/NR - frame structure," 2021. [Online]. Available: https://tinyurl.com/cwz8b6u9

[47] Verizon, "Verizon 5G standalone core trial paves way for robust 5G consumer and enterprise solutions," 2020. [Online]. Available: https://tinyurl.com/bddmb6ur

[48] M. R., "Low latency (not ultra) - A secret sauce of mmWave based 5G systems," Jan. 2021. [Online]. Available: https://tinyurl.com/2anj683d

[49] NetManias, "SK telecom 5G network architecture: 5G core, 5G RAN and 5G edge cloud (MEC)," Oct. 2019. [Online]. Available: https://tinyurl.com/ycx9m9d3

[50] AT&T, "5G coverage map," Jan. 2020. [Online]. Available: https://www.att.com/maps/wireless-coverage.html

[51] LGU, "5G coverage map," Jan. 2021. [Online]. Available: https://tinyurl.com/yxu96vhb

[52] Los Alamos National Laboratory, "Dynamic right sizing," 2001. [Online]. Available: https://public.lanl.gov/radiant/research/hpn/drs.html

[53] C. B. Samios and M. K. Vernon, "Modeling the throughput of TCP Vegas," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 2003, pp. 71–81.

[54] M. Polese et al., "milliProxy: A TCP proxy architecture for 5G mmWave cellular systems," in *Proc. 51st Asilomar Conf. Signals Syst. Comput.*, 2017, pp. 951–957.

[55] M. Zhang, M. Mezzavilla, J. Zhu, S. Rangan, and S. Panwar, "TCP dynamics over mmWave links," in *Proc. IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun.*, 2017, pp. 1–6.

[56] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *Proc. PFLDnet*, 2008, pp. 1–6.

[57] Qualcomm, "5G NR mmWave outdoor and indoor deployment strategy," May 2019. [Online]. Available: https://tinyurl.com/2ee5xwrx

[58] A. Narayanan et al., "Lumos5G: Mapping and predicting commercial MmWave 5G throughput," in *Proc. ACM Internet Meas. Conf.*, 2020, pp. 176–193.

[59] F. Huang, L. Tian, Y. Zheng, and J. Zhang, "Propagation characteristics of indoor radio channel from 3.5GHz to 28GHz," in *Proc. IEEE 84th Veh. Technol. Conf.*, 2016, pp. 1–5.

[60] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM Commun. Architectures Protoc.*, 1988, pp. 314–329.

[61] T. K. Herald, "Telcos establish 5G networks on nine subway lines nationwide," Jul. 2020. [Online]. Available: http://www.koreaherald.com/view.php?ud=20200723000584

[62] Google, "Get started with remote debugging Android devices," Aug. 2020. [Online]. Available: https://developer.chrome.com/docs/devtools/

[63] F. Li, J. W. Chung, X. Jiang, and M. Claypool, "TCP cubic versus BBR on the highway," in *Proc. 19th Int. Conf. Passive Act. Meas.*, Berlin, Germany, Springer, 2018, pp. 269–280.

[64] DASH Industry Forum, "dash.js," 2020. [Online]. Available: https://tinyurl.com/4n7sjdd8

[65] Akamai, "Big buck bunny," 2018. [Online]. Available: http://dash.akamaized.net/akamai/bbb_30fps/

[66] A. Zhou et al., "Learning to coordinate video codec with transport protocol for mobile video telephony," in *Proc. ACM 25th Annu. Int. Conf. Mobile Comput. Netw.*, 2019, Art. no. 29.

[67] J. Yi, M. R. Islam, S. Aggarwal, D. Koutsonikolas, Y. C. Hu, and Z. Yan, "An analysis of delay in live 360° video streaming systems," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 982–990.

[68] Google, "WebRTC," 2020. [Online]. Available: https://webrtc.org

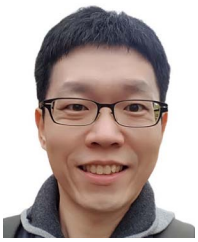
[69] G. Baig et al., "Jigsaw: Robust live 4K video streaming," in *Proc. ACM 25th Annu. Int. Conf. Mobile Comput. Netw.*, 2019, Art. no. 14.

[70] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "GLIMPSE: Continuous, real-time object recognition on mobile devices," in *Proc. 13th ACM Conf. Embedded Netw. Sensor Syst.*, 2015, pp. 155–168.

[71] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1421–1429.

[72] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proc. ACM Symp. Cloud Comput.*, 2018, pp. 401–411.
[73] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6230–6239.
[74] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.
[75] Y. Cui, Z. Lai, X. Wang, and N. Dai, "QuickSync: Improving synchronization efficiency for mobile cloud storage services," *IEEE Trans. Mobile Comput.*, vol. 16, no. 12, pp. 3513–3526, Dec. 2017.
[76] Ericsson, "How do you deploy 5G Core, and why must you do it now?," 2020. [Online]. Available: https://tinyurl.com/27chthdu
[77] ZDNet, "Nokia announces 5G 'standalone' network deployments for enterprises," 2020. [Online]. Available: https://tinyurl.com/yk7b7cp2
[78] Qualcomm, "Breaking the wireless barriers to mobilize 5G NR mmWave," May 2019. [Online]. Available: https://www.qualcomm.com/media/documents/files/5g-nr-mmwave-deployment-strategy-presentation.pdf
[79] S. Kang et al., "Fire in your hands: Understanding thermal behavior of smartphones," in *Proc. ACM 25th Annu. Int. Conf. Mobile Comput. Netw.*, 2019, Art. no. 13.
[80] A. Narayanan et al., "A comparative measurement study of commercial 5G mmWave deployments," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 800–809.
[81] A. Hassan et al., "Vivisecting mobility management in 5G cellular networks," in *Proc. ACM SIGCOMM Conf.*, 2022, pp. 86–100.
[82] R. Ford, M. Zhang, M. Mezzavilla, S. Dutta, S. Rangan, and M. Zorzi, "Achieving ultra-low latency in 5G millimeter wave cellular networks," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 196–203, Mar. 2017.
[83] A. Ichkov, P. Mähönen, and L. Simić, "End-to-end millimeter-wave network performance and mobility management overhead in urban cellular deployments with realistic pedestrian traffic and blockages," in *Proc. 18th ACM Symp. Mobility Manage. Wireless Access*, 2020, pp. 1–10.
[84] H. Haile, K.-J. Grinnemo, S. Ferlin, P. Hurtig, and A. Brunstrom, "Performance of QUIC congestion control algorithms in 5G networks," in *Proc. ACM SIGCOMM Workshop 5G Beyond Netw. Meas. Model. Use Cases*, 2022, pp. 15–21.
[85] H. Haile, K.-J. Grinnemo, P. Hurtig, and A. Brunstrom, "RBBR: A receiver-driven BBR in QUIC for low-latency in cellular networks," *IEEE Access*, vol. 10, pp. 18707–18719, 2022.

Hyoyoung Lim (Student Member, IEEE) received the BS degree from the Department of Information Science, Korea Military Academy, in 2009, and the MS degree from the Department of Information Science, Nagoya University, Japan, in 2015. She is currently working toward the PhD degree with the Department of Computer Science, University of Colorado Boulder. Her research interests include network performance measurement, wireless networks and systems, congestion control algorithms, and network security.



Jinsung Lee received the BS and PhD degrees in electrical engineering from KAIST, in 2003 and 2012, respectively. From 2012 to 2017, he was a senior engineer at 5G research lab in Samsung Electronics. From 2018 to 2021, he was a post-doctoral researcher at the Department of Computer Science, University of Colorado Boulder. He is currently a senior staff engineer with Qualcomm Technologies, Inc. His research interests include wireless networks and systems, live video streaming, low-latency transport protocols, and mobile deep learning. He received the Best Paper Awards from IEEE SECON, in 2013 and ACM MobiSys, in 2019.

Jongyun Lee received the BS and MS degrees in computer science and engineering from UNIST, in 2018 and 2020, respectively. He is currently working toward the PhD degree with the Electrical and Computer Engineering Department, Seoul National University. His research interest includes energy-efficient edge computing and low-latency metaverse applications.

Sandesh Dhawaskar Sathyanarayana received the MS degree from the Department of Computer Science, University of Colorado Boulder, in 2019, and is currently working toward the PhD degree with the Department of Computer Science, University of Colorado Boulder. His research interests include mobile systems and transport protocols.

Junseon Kim received the BS degree in electronic engineering, the MS degree in electrical, electronic, and control engineering from Hankyong National University (HKNU), Anseong, Gyeonggi, Republic of Korea, in 2014 and 2016, respectively, and the PhD degree in computer science from the Ulsan National Institute of Science and Technology (UNIST), Ulsan, Republic of Korea, in 2023. He is currently a post-doctoral researcher with the Department of Electrical and Computer Engineering, Seoul National University (SNU). His research interests include latency-guaranteed networking (with ML) for next-generation cellular networks (i.e., 5G-Adv/6G).

Anh Nguyen received the BS degree in computer science from the University of Science, VNU-HCMC, Vietnam, in 2009, the MS degree in software engineering from Chonnam National University, South Korea, in 2012, and the PhD degree in computer science from the University of Colorado Boulder, in 2022. She is currently an assistant professor with the Department of Computer Science, University of Montana and leading the Mobile Cyber-Physical Intelligence Lab. Her research interests span developing novel sensing and intervention technologies for smart health, the Internet of Things, and human-computer interactions. She received ACM SenSys Best Paper Award, in 2016.

Kwang Taik Kim (Senior Member, IEEE) received the BS degree in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2001, and the MS and PhD degrees in electrical and computer engineering from Cornell University, Ithaca, New York, in 2006 and 2008, respectively. He is a research assistant professor with the Elmore Family School of Electrical and Computer Engineering, Purdue University. His research interests include communication engineering, open architecture, edge platform, and large-scale distributed computing. He was a recipient of the 2014 Samsung CEO Award of Honor with the Technical Division, the 2016 Samsung Best Paper Award (Gold Prize) with the Communication and Network Division, and the Best Paper Award with 2021 ACM MobiHoc. He serves as an associate editor for the *IEEE Networking Letters*.

Youngbin Im (Member, IEEE) received the BS and PhD degrees in computer science and engineering from Seoul National University, in 2006 and 2014, respectively. He was a post-doctoral researcher with the Department of Computer Science, University of Colorado Boulder, from 2015 to 2019. He is currently an assistant professor with the Department of Computer Science and Engineering, UNIST. His research interests include the next-generation internet, video streaming, internet protocols, data centers, wireless networks, and the IoT. He received the Best Paper Award from ACM MobiSys, in 2019.

Mung Chiang (Fellow, IEEE) is the president of Purdue University and Roscoe H. George distinguished professor of electrical and computer engineering. Previously he was Arthur LeGrand Doty professor of electrical engineering with Princeton University. He received the Alan T. Waterman Award, Guggenheim Fellowship, IEEE Infocom Achievement Award, IEEE Kiyo Tomiyasu Award, and is a member of National Academy of Inventors and Royal Swedish Academy of Engineering Sciences.

Dirk Grunwald (Senior Member, IEEE) received the PhD degree in computer science from the University of Illinois, Urbana-Champaign. He is a professor with the Department of Computer Science, University of Colorado Boulder. His research focuses on computer systems, broadly defined.

Kyunghan Lee (Member, IEEE) received the BS, MS, and PhD degrees from the Department of Electrical Engineering, KAIST (Korea Advanced Institute of Science and Technology), Daejeon, South Korea, in 2002, 2004, and 2009, respectively. He is currently an associate professor with the Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea where he is leading the Networked Computing Lab. He had been with the School of Electrical and Computer Engineering, UNIST, Ulsan, South Korea from 2012 to 2020. He received IEEE ComSoc William R. Bennett Prize, in 2013 and 2016, respectively, and received the best paper award from ACM MobiSys, in 2021. He is serving on the editorial board of *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Vehicular Technology*, and *Computer Networks*. His research interests include performance-guaranteed networking and thermally-reliable mobile machine learning.

Sangtae Ha (Senior Member, IEEE) received the PhD degree in computer science from North Carolina State University. He is an associate professor with the Department of Computer Science, University of Colorado Boulder. He was an associate research scholar with Princeton University from 2010 to 2013. He received the Best Paper Awards from ACM MobiSys, in 2019 and 2021.