Octopus: Exploiting the Edge Intelligence for Accessible 5G Mobile Performance Enhancement

Congkai An, Anfu Zhou, Jialiang Pei, Xi Liu, Dongzhu Xu, *Student Member, IEEE*, Liang Liu, and Huadong Ma, *Fellow, IEEE*

***Abstract*— While 5G has rolled out since 2019 and exhibited versatile advantages, its performance under high/extreme mobility scenes (*e.g.*, driving, high-speed railway or HSR) remains mysterious. In this work, we carry out a large-scale field-trial campaign, taking >13,000 Km round-trips on HSR moving at 250-350 Km/h, with operational 5G cellular coverage along the railway. Our empirical study reveals that coupling interaction among high mobility, 5G handover characteristics, and applications' sluggish reaction to handover, results in catastrophic damage to user experience: low TCP bandwidth utilization of 26.6% and glitchy 4K VoD streaming. To solve the problem, we propose an edge-assisted mobility management framework called Octopus. Different from previous works, Octopus aims at a standard-compatible and easy-to-deploy solution, thus we take a new design paradigm of exploiting the edge intelligence on multi-access edge computing (MEC). We realize Octopus as a universal MEC service ready for benefiting any third-party mobile applications. We prototype, deploy, and evaluate Octopus in operational 5G, which demonstrates the significant performance gain across the full-range mobile scenarios, *e.g.*, HSR, driving, and walking.**

***Index Terms*— 5G, mobility management, transport protocols, edge computing.**

I. Introduction

5G, WHILE still at an early stage, has already manifested tremendous impact. With multi-Gbps bitrate and ultra-low network latency [1], [2], 5G is stimulating various innovations, from virtual/augmented reality [3], volumetric video streaming [4], [5] to the cutting-edge metaverse [6]. Furthermore, 5G is also going beyond to transform a number of vertical industries [7], [8].

However, the standard performance specifications of 5G are mainly benchmarked under static or low-mobility scenarios, while the high/extreme mobility cases remain mysterious. As a long-standing issue traced back to 3G/4G ages, the mobile performance degradation is becoming more and more disconcerting in the 5G era for two reasons. *(i) Applications going highly mobile*. By December 2021, 28 countries have developed high-speed-railway (HSR) to connect major cities [9]. Among them, China has built over 40,000 Km HSR at ∼300 Km/h speed, which serves >2,290 million passengers per year [10], much more than 660 million airline throughput [11]. In addition, as autonomous vehicles mature, people envision highly mobile applications on vehicles or HSR including mobile office (*e.g.*, video/hologram conference), entertainment (*e.g.*, 4K/8K video and gaming), and online education, *etc*. *(ii) Smaller 5G cell service range*. Measurement studies [12], [13], [14] show that the Sub-6 GHz 5G cell coverage is only 19.56% of 4G cell and mmWave 5G cell is even lower, which leads to more frequent cross-cell handover.

To answer the concern, we perform a large-scale measurement campaign to quantify and demystify 5G performance under extreme mobility. Unlike recent studies [12], [14], we deliberately spotlight HSR,[1] so as to focus on the impact of extreme mobility, while simultaneously isolating other factors, *e.g.*, irregular coverage, traffic congestion, which usually happens in urban areas. The campaign takes >13,000 Km round-trips on the HSR, and accumulates 730 GB traces across one year. Our measurement reveals that: *(i)* Extreme mobility impacts 5G much more severely, compared with its 4G counterpart. In particular, 5G has 3× more frequent handover (HO) and 1.87× longer duration per HO on average. Besides the smaller cell size, we uncover other reasons behind, such as more dynamic channels and the non-standalone (NSA) deployment model. *(ii)* The impact is also amplified when propagated to upper-layer protocols/applications, *e.g.*, a low TCP bandwidth utilization of only 26.6%, far less than 51.1% of that in 4G, and glitchy 4K VoD streaming. Moreover, we find a mapping relationship between HO frequency and application QoE, from which intensive 5G handover under extreme mobility results in significant QoE degradation. Therefore, ensuring seamless connectivity for the massive highly/extremely mobile users is a crucial issue for 5G.

In this paper, we aim to address the problem in an accessible way, *i.e.*, compatible with the legacy cellular framework and easy to deploy. We note that, despite a longitudinal research effort on this topic along with 3G/4G/5G evolution, they

Manuscript received 24 April 2022; revised 18 September 2022; accepted 19 November 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. Peng. This work was supported in part by National Key Research and Development Program of China under Grant 2019YFB2102202, in part by the Innovation Research Group Project of NSFC under Grant 61921003, in part by the International Cooperation and Exchange of the National Natural Science Foundation of China under Grant 61720106007, in part by NSFC under Grant 61832010, in part by the 111 Project under Grant B18008, and in part by the Youth Top Talent Support Program. *(Corresponding author: Anfu Zhou.)*

The authors are with the Beijing Key Laboratory of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: ack@bupt.edu.cn; zhouanfu@bupt.edu.cn; peijl@bupt.edu.cn; xi_1998@bupt.edu.cn; xdz9601@bupt.edu.cn; liangliu@bupt.edu.cn; mhd@bupt.edu.cn).

Digital Object Identifier 10.1109/TNET.2022.3224369

[1] A 174 Km HSR from Beijing to Zhangjiakou, the major host cities of the 2022 Winter Olympics. More than 200 5G base stations are deployed along the railway, which results in perhaps the longest HSR segment with continuous 5G coverage.

are hard to deploy, *e.g.*, tailoring specific upper-layer protocols [15], [16], re-factoring legacy cellular protocols [17], [18], or cross-layer optimization on the UE side [19], [20]. They involve heavy system revision and require root/system privileges, which are not applicable to either legacy cellular framework or off-the-shelf UE devices.

To bridge the gap, we propose Octopus, a 5G native mechanism that enables the upper layers to adapt to HO promptly, so as to salvage application QoE even under extreme mobility. To meet the "accessible" requirement, the key idea of Octopus is to embrace the growing edge intelligence in 5G.[2] In particular, Octopus runs at the multiple-access edge computing (MEC) server behind the 5G base station (gNB), as a MEC service. It continuously monitors the cellular control channel and intercepts HO events, upon which it notifies the transport layer or application layer of the application server *explicitly*. Then they can make adaptions to HO, so as to shield HO's adverse effect, thus improving upper layers' performance.

As an edge service, Octopus is lightweight, which only involves software-level changes on MEC and backend application servers, and evades hardware/firmware changes and root/system privilege on the UE/gNB, unlike existing solutions [20], [21], [22]. Meanwhile, Octopus is a pioneering and feasible solution for future cellular NEF (Network Exposure Function) on seamless 5G mobility management. We emphasize that the recent 5G network specifications have supported the popular NEF approach in the operational systems [23], [24], [25], [26]. Driven by this, the application servers in the near future will be customized and actively adapt to NEF to improve user QoE, *i.e.*, the deployment overhead brought by Octopus to the backend servers will be smaller and smaller. Hence Octopus is ready to be widely deployed, particularly considering that 5G operators are just starting to update 5G from non-standalone (NSA) mode to SA mode and deploy MEC servers for the advanced 5G [27], [28], [29]. In addition, Octopus also has a significant advantage in terms of reducing the response latency to HO events (Sec. III-B). The operation flow of Octopus involves signaling and coordination among gNB, MEC, and the application server. To make Octopus standard-compatible and lightweight, we further harness the endogenous capabilities of the 5G MEC framework. Octopus incorporates the standard Radio Network Information Sevices (RNIS) interface [26] in the MEC platform to intercept each HO event reliably. Furthermore, for transport layer, Octopus reuses the TCP connection between the mobile UE and the server, and enables a reserved bit in the packet header to inform the event; for application layer, we utilize a socket-based coordination mechanism to pass up the notification.

To demonstrate Octopus's universality as a MEC service, we apply it to two use cases, *i.e.*, BBR-H (enhancing TCP BBR [30]) and DASH-H (enhancing video streaming DASH protocol [31]) in Sec. IV. With Octopus's timely HO notification, they can distinguish the normal network fluctuation from HO-induced illusion, and thus avoid or remedy the interference from HO. For instance, BBR-H re-configures its system parameters for faster recovery upon HO; and DASH-H supervises the video bitrate decisions and vetos inaccurate ones by examining the decision/HO time series. Notably, there involve only 6 lines of C code and $\sim$30 lines of Python/NGINX code modification for BBR and DASH use cases respectively, which demonstrates Octopus's lightweight for upper layers.

We implement a full-fledged Octopus prototype system, consisting of 5G smartphones, gNBs (USRP as radio module), a MEC server, and a remote application cloud server. We also perform extensive field tests of Octopus on HSR, and the experimental results show that *(i)* Octopus can deliver UE's HO events to the cloud server in real time, which translates into remarkable performance gains: enhancing the BBR throughput by 2$\times$ on average across all trials (Sec. V-B), and DASH video bitrate by more than 60%. Moreover, DASH bitrate oscillation and stalling rate are reduced by more than 65.9% and 30.4%, respectively (Sec. V-C). *(ii)* As a MEC-assisted mobility management framework, Octopus not only benefits the extreme-mobility HSR scenario, but also helps more common cases like driving (improving BBR throughput by 27.9%, and DASH video bitrate by 31.7%) and walking (BBR throughput increased by 13.6%, DASH video bitrate increased by 14.5%).

To our knowledge, Octopus represents the first edge assisted mobility management framework ready to be widely deployed. The contributions can be summarized as follows:

(i) We perform a large-scale field trial of operational 5G under HSR, which presents the surprisingly poor performance. The further analysis uncovers the root reason for the sluggish reaction to handover in existing cellular architecture.

(ii) We design Octopus, an edge-assisted mobility management framework, which runs as a MEC service to enable seamless mobility in an easy-to-deploy and efficient way.

(iii) We prototype and evaluate Octopus in operational 5G networks, which leads to significant performance gain in extensive mobile scenarios.

II. Demystifying 5G Performance Under High Mobility

Though cellular mobility has been studied extensively [15], [16], [32], [33], [34], our measurement focuses on two issues yet to be known: *(i)* Compared with 4G, what are the different handover characteristics of the emerging 5G, particularly under the most challenging HSR scenario? *(ii)* How severely does 5G handover impact upper-layer application performance *quantitatively*, despite the qualitative conjecture inherited from 3G/4G era? Moreover, we are interested in the underlying interaction between HO events and the behaviors of modern upper-layer protocols. We answer the questions by large-scale field trials, starting with measurement setup (Sec. II-A), and then proceeding to mobility statistics (Sec. II-B), application performance degradation (Sec. II-C).

A. Measurement Setup and Dataset

Experimental scenarios. We carry out our measurement mainly on the 174 Km HSR with full Sub-6 GHz 5G coverage by 217 5G gNBs. Some of these 5G base stations (gNBs) are co-located with 4G base stations (eNBs), following the NSA-3a EN-DC [35] network deployment model, which is a common practice widely used by operators to reduce

[2]Octopus is well-known for strong "edge intelligence". 60% of an octopus's neurons are on its tentacles.

TABLE I
DATASET DESCRIPTION ON 5G MOBILITY MEASUREMENT STUDY IN THE WILD

Speed Type	Scenario	Speed (Km/h)	Mileage (Km)	Traces (GB)		RRC Signalings	HO Times	Flows (GB)			DASH Traces (MB)
				4G	5G			TCP	UDP	DASH	
High	HSR	250-350	13984	60	670	574101	19562	306.7	64.3	273	80.0
Medium	Driving	60-120	270	-	96	22181	765	86.0	-	10	3.5
Low	Walking	4-5	18	-	63	2226	75	45.3	0.8	17	1.1

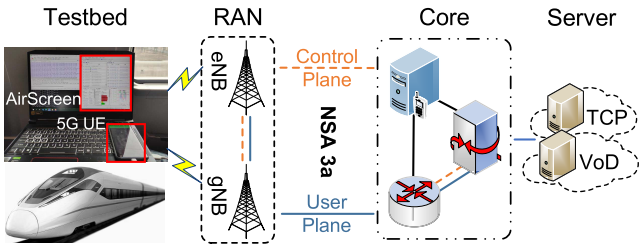

Fig. 1. Measurement platform.

TABLE II
HO STATISTICS AT PHYSICAL LAYER

Parameter		Value	
		4G	5G
Interval	Average HO interval (s)	20.22	6.65
Overhead	HO time ratio (%)	0.27	1.04
	Average HO duration (ms)	53.6	92.8
	Data packet loss ratio (%)	0.01	0.02
Failure	HO failure ratio (%)	7.33	12.7

deployment cost. Therefore, we can compare the performance differences between 4G and 5G, in a fair way. In addition, we have also extended our measurement study to other mobile scenes, including walking (4-5 Km/h) on a university campus and driving (60-120 Km/h) on urban roads.

Measurement devices. As shown in Fig. 1, we tether a 5G Android UE (ZTE Axon10 Pro, a typical 5G smartphone with powerful SDX M50 5G modem, snapdragon TMM855 CPU, and 256 GB storage, which can also access the 4G networks when turning off the 5G option) to a laptop (Acer N20C1) with a USB 3.0 cable. So the laptop acts as a 5G client and relies on the UE for Internet access. In this way, we can develop applications and measurement tools more freely on the laptop, without the limitation of Android security protection. Our measurement of upper-layers' performance is mainly carried out over the path between the client and a high-performance cloud server (Ubuntu 18.04|8vCPUs|64GB). The server is provisioned with abundant networking capacity, *i.e.*, 1Gbps bandwidth, to accommodate the ultra-high-speed 5G traffic.

Measurement tools and methodology. We use a variety of tools for cross-layer measurement. *(i)* For physical 5G cellular radio monitoring, we equip the client with a 5G debugging tool, AirScreen [36]. In detail, AirScreen runs on the laptop and monitors the UE's diagnostic interface, which solves the problem that we cannot directly obtain UE's movement information from the BS-side in the current 5G network, *i.e.*, AirScreen can obtain the 5G RRC/physical-layer signaling messages (*e.g.*, handover trigger event, association events, and so on) and channel QoS parameters (*e.g.*, signal strength, connection service status, and access log) from the local UE (UE-centric monitoring approach). *(ii)* For transport-layer measurement, we use iperf3 [37] to measure TCP metrics under the control of two representative TCP algorithms, *i.e.*, CUBIC [30] and BBR [38]. Meanwhile, we adopt Wireshark [39] to capture all packet traces received by the client, and log the performance indicators ⟨*CWND, RTT, bottleneck bandwidth, pacing rate*⟩ of the TCP connection from the server kernel. *(iii)* For application-layer measurement, we deploy a video streaming server (NGINX 1.19.8 [40]), using the widely-used *dash.js* framework [31], [41]. Meantime, we run a DASH streaming client on the laptop (the max buffer size is 30 seconds like YouTube), which requests and playbacks video segments via 5G access. The DASH VOD system includes three prevalent ABR algorithms, which perform video bitrate adaptation based on historic throughput, video buffer-level, or both, for better QoE [42]. For a fair comparison, we use the same 20-minute video file for each measurement, and split it into a sequence of 1.6-second-long chunks with multiple resolutions (*i.e.*, {360P, 720P, 1080P, 2K, 4K}) via FFmpeg [43] and Bento4 [44].

Dataset. Our measurement spans three scenarios (*i.e.*, HSR, driving, and walking), and accumulates 889 GB traces over 14,272 Km trips during 2020∼2021. As summarized in Table I, we record 598,508 radio resource control (RRC) signalings, consisting of 20,402 HO-related events, *i.e.*, HO triggering, execution, completion, and also the logical signaling behind. We also capture transport-layer traces of TCP and UDP flows (438.0 GB and 65.1 GB respectively), and log the throughput, congestion window, packet loss, and RTT metrics. For the application layer, we play a total of 300 GB of videos and collect 84.6 MB DASH control message traces. For a comprehensive comparison, we also collect 4G traces, by repeating the same experiments while disabling 5G on the smartphone. *We are now organizing our datasets and tools and will release them for facilitating future studies.*

B. 5G Handover Characteristics on HSR

Table II summarizes the statistics of all HO events across all the HSR trips, from the physical layer perspective.

Handover interval. We find that the average 5G HO interval (*i.e.*, the time gap between two consecutive HO events) is 6.65s, when traveling on the HSR at 250∼350 Km/h. Compared with the 4G case, the 5G HO interval is much shorter, only $\frac{1}{3}$ of that of 4G, since the 5G coverage radius is much smaller. A surprising observation is that *a large fraction of HO events happen within a very short interval, e.g., 35.5% intervals are less than 2 seconds* (Fig. 2). By looking into the trace, we find that the reason lies in a *"ping-pong" effect*,

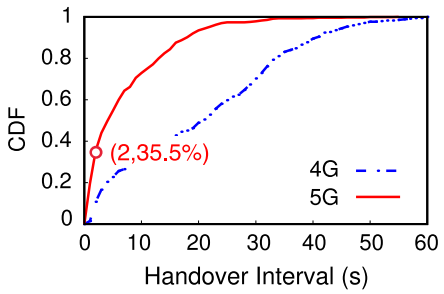

Fig. 2. HO interval statistics.

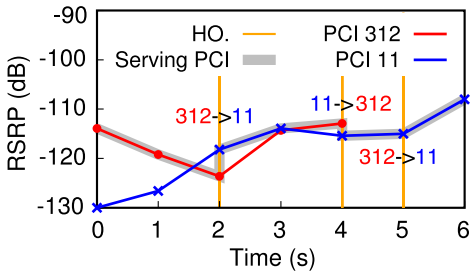

Fig. 3. "Ping-pong" effect of 5G HO.

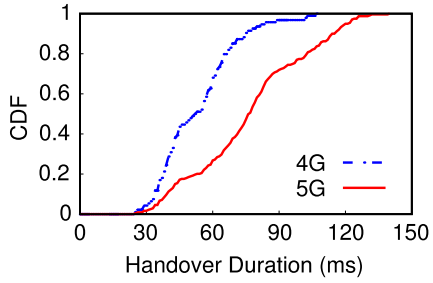

Fig. 4. HO duration.

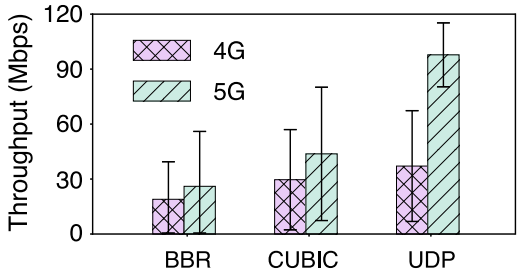

Fig. 5. Throughput comparison.

i.e., a UE may switch back-and-forth between two adjacent cells. The finding is unexpected because 5G gNBs are regularly distributed along the railway, and users inside a train move forward in the same direction, unlike the urban scenarios where a gNB's coverage is irregular, and users may move forward and backward freely. To demystify the problem, we carefully observe the changes in 5G signal strength (*i.e.*, RSRP, the metrics of triggering HO decisions in 4G/5G [18]) when the UE passes through any two adjacent gNB cells. We find that the 5G RSRP occasionally exhibits high fluctuation, instead of following a monotonic trend as the train moves. As a showcase, Fig. 3 plots a 6s-length RSRP time series, in such a short period, the RSRP handover threshold set by RAN can be considered unchanged. We can observe that the RSRP of gNB cell 312 and cell 11 continuously and greatly jitter, which misleads the UE's handover decision and leads to "ping-pong" HO, even under the simplistic HSR scenario. Moreover, we analyze the results of multiple HSR trips, the statistics show that "ping-pong" handovers account for about 19.3% of all handovers and have a poor correlation with factors such as weather conditions and physical location.

Handover overhead. We first calculate the HO duration, defined as the time gap between a handover initiation and its completion (Fig. 4). We use AirScreen to capture and extract the HO signaling, starting with *RRC_Reconfiguration* (carrying new gNB parameters) and ending with *RACH_success*, to time a HO following the standard [45]. During a HO, a UE is busy handling the HO control procedure (*i.e.*, such as scanning neighboring cells, de-associating with the current gNB and associating with the new gNB), which interrupts normal data transmission [46], and may lead to packet loss or long RTT from the upper layers' perspective. We make a few observations from Table II results: *(i) The duration of a 5G HO is 92.8 ms on average, much longer (i.e., 1.73×) than that of 4G. The reason lies in the NSA mode deployment in current 5G [12], where the gNB reuses legacy 4G cellular core for control plane operation including HO management, and thus it incurs extra 4G control operations and longer HO duration. (ii)* Despite the relatively long duration, the 5G HO is still accomplished quite efficiently: the sum of HO duration accounts for only a very small fraction of the whole experimental time (*i.e.*, 1.04%). *(iii)* Moreover, the overall packet loss rate (calculated from TCP connections) is also very low, only 0.02% for 5G and 0.01% for 4G (excluding out-of-order packets), as the gNB and eNB usually have a large buffer to hold the packets arriving during a HO [47].

Handover failure. A 5G HO failure may happen due to unsuccessful link synchronization, or poor coverage. The failure may lead to a link outage, or force the UE to fall back to 4G access, thus causing severe performance degradation. In 5G HSR scenario, HO failures account for around 12.7% of all HO events. Anyway, the majority of 5G HO events still succeed, and we focus on their characteristics and impact.

C. 5G Handover Impacts on Higher Layers

1) TCP Performance: We evaluate two TCP variants: *(i)* CUBIC, as Linux's default TCP algorithm, uses packet loss as the congestion metric [48]; *(ii)* BBR, a representative of the new generation TCP, explicitly probes network bandwidth. Previous work [12] verified that BBR outperforms CUBIC in *static* 5G networks. Our experiments log the performance statistics of both across the HSR scenario (Fig. 5).

Overall performance. We observe that *(i)* the UDP baseline performance drops significantly on HSR. The average throughput is only 97 Mbps, about $\frac{1}{2}$ of static links due to the high channel dynamics in high mobility (the UDP throughput is about 200 Mbps and 280 Mbps when inside and outside the stopped train respectively, due to the shielding effect of the train's metal compartment.[3]) *(ii)* TCP performance degrades

[3]Note that here the static throughput is much lower than that in [12], due to different measurement positions and cellular operator's configurations.

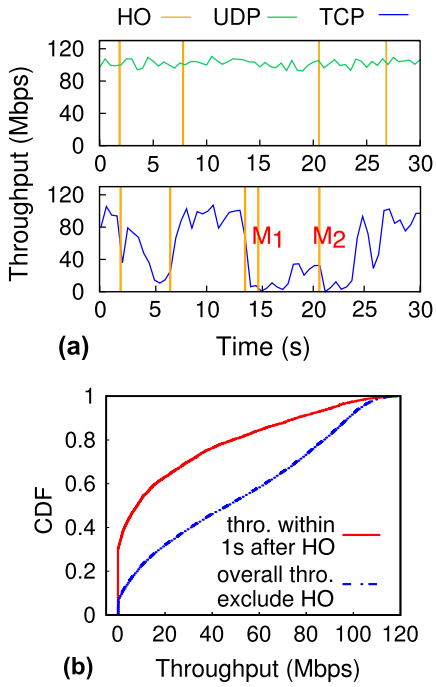

Fig. 6. Harm of 5G HO to CUBIC.

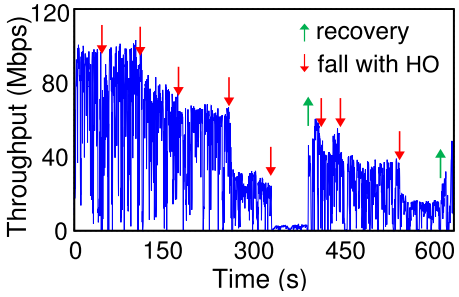

Fig. 7. Stepwise decline and recovery in BBR throughput.

TABLE III
BITRATE DETAILS OF EXPERIMENTAL VIDEO SOURCE

Resolution	4K	2K	1080P	720P	360P
Avg. Bitrate (Mbps)	26.8	12.0	7.2	3.6	1.2

severely, approaching only 26.6% and 44.7% of the UDP baseline, for BBR and CUBIC respectively. *(ii)* More interestingly, the relative performance of CUBIC and BBR is opposite to that in the static scenario. For static 5G cases, CUBIC's utilization ratio against UDP is only 31.9%, whereas BBR achieves a much higher utilization of 82.5% [12]. However, *in the HSR scenario, BBR suffers more from frequent HO events.* For mobile 4G, the relative performance still holds albeit with a lower absolute throughput value. In the following, we proceed to uncover the underlying reasons.

CUBIC. The snapshot measurement in Fig. 6 (a) shows how TCP throughput evolves during high mobility. We observe that *(i)* The CUBIC throughput decreases dramatically upon HO events, down to 0 in certain cases (*e.g.*, timestamps M_1 and M_2). In contrast, UDP throughput is rarely impacted by HO and remains stable, which explains their throughput gap. *(ii)* Moreover, *the impact of HO on CUBIC persists much longer than the HO completion time*. Specifically, the CUBIC throughput usually remains low for 500 ms after the HO, and sometimes up to 3s, whereas the HO itself takes only around 92.8 ms to complete at the physical layer (Table II). We analyze CUBIC traces and find that HO events cause TCP CWND to fail to increase until all delayed ACK packets are received.

We find that the TCP throughput degradation attributes to HO, but *not* to channel fluctuation during mobility. In particular, we compare the throughputs immediately after each HO (1s time window), against the remaining in Fig. 6 (b), which clearly shows that the former ones are far lower.

BBR. We showcase a snapshot of BBR's throughput evolution in Fig. 7. During consecutive HO events, BBR throughput gradually drops, and remarkably, it cannot recover promptly from such drops. Instead, the throughput keeps decreasing step by step (we call it "throughput fall" as marked by ↓), and starts to recover (marked as ↑). We find that BBR will have such a stepwise decline in about 75 seconds, but it takes more than 180 seconds to trigger its "Startup" stage (in a gain of 2/ln2 to increase the sending rate exponentially [30]) to recover. *Due to such a sluggish response to HO, BBR's throughput is much less than that of CUBIC.*

We proceed to detail BBR operation mechanism and find the root factor causing "throughput fall" is the inaccurate estimation of $B_{tl}BW$. BBR executes congestion control by estimating the network path capacity based on two parameters: RT_{prob} (round-trip propagation time) and $B_{tl}BW$ (bottleneck bandwidth), the product of which determines the congestion window size (CWND). Formally, $B_{tl}BW$ is derived as:

$$\widehat{B_{tl}BW}_T = \max\{\frac{\Delta delivered_t}{interval\_us_t}\}, \forall t \in [T - W_B, T], \quad (1)$$

where $\Delta delivered$ represents the amount of data to acknowledge upon an ACK packet, *interval_us* represents the time interval since the last $\Delta delivered$ is calculated, and the time window W_B is 10 RTTs. Normally, *interval_us* is very low (58.2% values $\leq$ 100 ms). But HO will cause a large number of delayed ACK packets, which escalates the estimated *interval_us* by several times, much larger than the actual value. This in turn causes severe underestimation of $B_{tl}BW$.

2) UHD Video Streaming: We now investigate whether 5G can support 4K UHD video streaming under high mobility. Table III lists the bitrate statistics in our experimental DASH VoD system (Sec. II-A). As we can see, the most demanding 4K video entails a bitrate of 26.8 Mbps on average. At first blush, this is much lower than the average throughput 43.7 Mbps of CUBIC (the TCP protocol below DASH), thus for DASH application typically with long buffers (30 seconds in our case), 5G should be able to support smooth 4K streaming.

However, the result in Fig. 8 contradicts this intuition. 4K video only takes 37.44% of the whole playback time, worse still, the video streaming even freezes for 2.27% of the playback time. On the other hand, we compute the downloading throughput of DASH video segments and plot the CDF in Fig. 9, we find that 64.5% of the segments have a throughput exceeding 26.8 Mbps. The results imply that the *HO events interfere with DASH's ABR algorithm, and mislead it to choose too many low-bitrate video segments, even when higher*

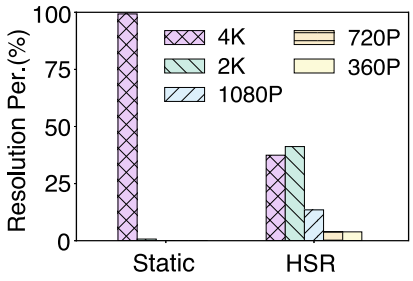

Fig. 8. DASH video resolution in static and HSR scenarios.

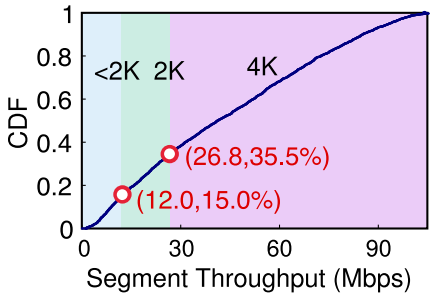

Fig. 9. CDF of DASH segments' throughput.

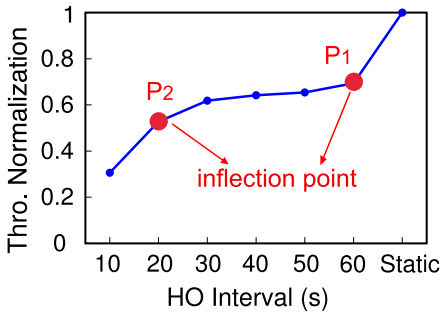

Fig. 10. Effect of HO interval on BBR.

bitrates could be supported. The reason is that the HO causes a sharp but instantaneous decline of throughput or buffer-level, which makes the ABR algorithm believe that the network condition becomes very poor, thus making over-conservative choices, *i.e.*, requesting low-bitrate video segments (more details will be discussed with a showcase in Sec. V-C). *Worse still, we find that DASH on HSR needs to take at least 10s to recover (to avoid the severe video bitrate oscillation, the ABR algorithm limits its growth rate in the next 8 decisions [42]), which simply cannot afford the frequent 5G HO events (i.e., 6.65s per HO) and finally leads to such poor QoE.*

D. Mapping Between Handover Frequency and Application Performance

We perform a micro-benchmark field trial to investigate the impact of UE mobility. In particular, we deliberately choose a boundary point between two gNB cells, and move the UE device back and forth to reduce RSRP so as to trigger HO events. We repeat the experiments for 2.25 hours, at different walking speeds, which lead to different inter-HO intervals. We measure the BBR throughput in each experiment and plot the normalized result in Fig. 10. We observe that BBR

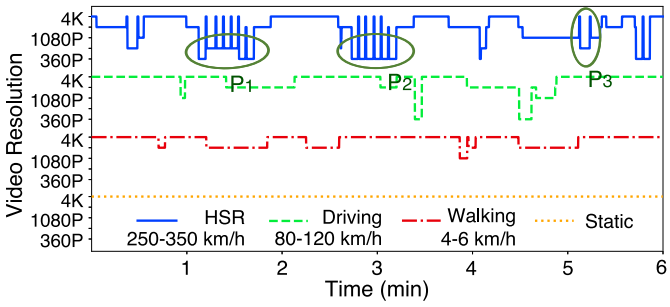

Fig. 11. DASH's convergence in multiple scenarios.

throughput constantly drops with decreasing HO interval, non-linearly. In particular, there are two inflection points: one lies at P_1 corresponding to the 60s HO interval, which indicates throughput drop once a UE changes from static to moving; the other one, more interestingly, lies at P_2 corresponding to the 20s HO interval, which is exactly the boundary between 4G and 5G's HO frequency in the HSR scenario (Table II). The observation corroborates the accelerated destructive effect of 5G handover in HSR scenarios. We have a similar observation for DASH VoD. Fig. 11 showcases DASH video resolution changes under different mobile scenarios (static, walking, driving, HSR), from a segment of field experiment trace in Table I dataset. We find that under the HSR scenario, *the most frequent HO events make DASH unable to converge to a stable bitrate*, but keep oscillating among successive HO events, as marked in P_1, P_2, P_3.

Summary: The measurement campaign reveals that more frequent HO of emerging 5G under high/extreme mobility, coupled with applications' sluggish reaction to HO, results in unprecedented damage to upper layers' performance, particularly for protocols like TCP and DASH which make critical decisions based on the historical network conditions measured at end hosts. The observation motivates us to transcend the existing solutions that are hard to deploy, but to explore accessible solutions.

III. Octopus Design

A. Overview

It is very challenging to design an easy-to-deploy mobility management solution, as it involves multiple parties, including UE (restricted OS or firmware changes), gNB (completely closed RAN management), and backend servers. Fortunately, the emerging 5G is going open with many inter-operable interfaces and MEC to enable software-defined innovations [25], [49], [50]. By leveraging this trend, we propose Octopus, a MEC service that exploits the edge intelligence for seamless mobile 5G. Octopus sits behind cellular RAN and operates in two steps as shown in Fig. 12: *(i)* Octopus keeps monitoring the control channel of cellular networks and detects any HO events in its domain. *(ii)* Upon a HO, Octopus can notify the server, which will adjust its transport-layer or application-layer actions, as required, so as to improve resilience to HO. Octopus must address two key challenges as follows:

Standard compatibility (Sec. III-B). 5G is extremely complex, comprised of hundreds of standard protocols, in which more than 40 protocols are involved in mobility management [51]. To mitigate the HO impacts, a straightforward

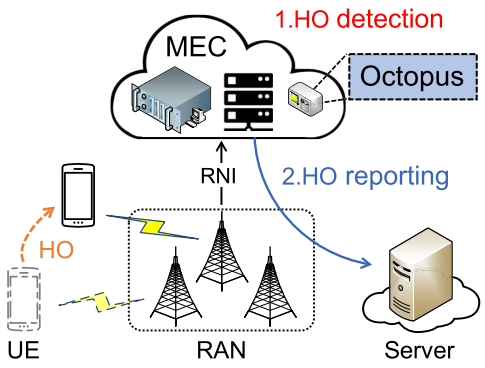

Fig. 12. Octopus overview.

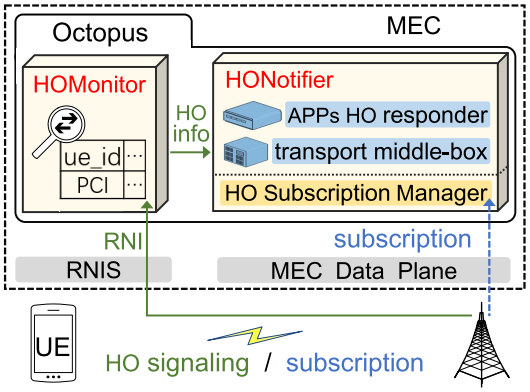

Fig. 13. Octopus at MEC.

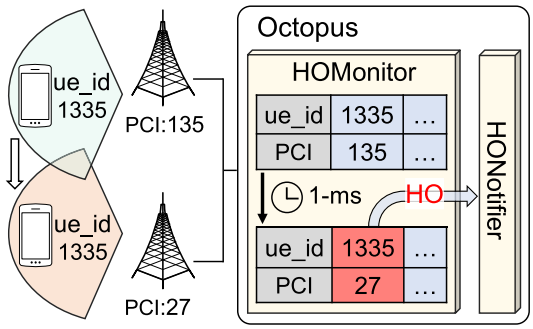

Fig. 14. HO analyses in Octopus.

idea is to revise these mobility management protocols, and allow them to inform the upper layers explicitly. However, this will not be compatible with the legacy system that has already been widely deployed, *i.e.*, over 1.2 million 5G gNBs worldwide [52]. In addition, it breaks the modularity of the network stack, since the 5G RAN is managed by the cellular operators whereas the upper layers are manipulated by application service providers. Therefore, we design Octopus on the top of 5G MEC framework, a unique vantage point allowing the two to interact. On the one hand, Octopus leverages the radio network information service (RNIS) capability of MEC to capture HO events at the RAN level, following the 3GPP NEF (Network Exposure Function) and ETSI (European Telecommunications Standards Institute) standards [23], [24], [25], [26]. On the other hand, Octopus as a MEC service is registered into the list of services on the MEC platform [25], to benefit various application providers.

Lightweight (Sec. III-C). To facilitate large-scale use in the real world, Octopus needs to simplify the deployment and reduce the resource overhead, including computing and network resources. Therefore, Octopus builds a lightweight HO subscription and feedback mechanism. To feedback HO to server transport layer, it operates like a *middle-box*, which utilizes the ongoing TCP connection between the local UE and remote server, and lets the in-band TCP packet piggyback the HO events. As for server application layer, we design a socket-based coordination mechanism to share the HO information received by the server transport layer with its application layer.

In what follows, we proceed to two key modules of Octopus: HO monitoring (Sec. III-B) and notification (Sec. III-C).

B. Handover Monitoring

Strawman approaches. A straightforward idea is to directly monitor UE's HO through some existing dedicated tools (*e.g.*, XCAL [53], AirScreen [36], MobileInsight [21]) on mobile devices. However, off-the-shelf devices usually do not open the interface to expose the UE's underlying information (*e.g.*, gNB's RSRP, RRC messages), considering the security and stability of UEs, so these tools all need to root UE to obtain its system access privilege. From [54], a very small fraction (only 7.5%) of global mobile devices are rooted, which hinders its applicability in practice.

Octopus's MEC-centric design. Octopus aims to capture HO events accurately and efficiently from gNB side so as to maintain compatibility and modularity. To this end, we design *HOMonitor*, a MEC service based on RNIS, as shown in Fig. 13. In general, the functions of MEC include "MEC Applications" and "MEC Services". The former represents applications that sink to the edge, *e.g.*, virtual/augmented reality, V2X [3], [55]; the latter is a number of modules that provide related services to applications in MEC (after the applications register on the MEC platform). For instance, RNIS can directly obtain 5G RAN's radio network information (RNI), *e.g.*, IMSI, PCI, RSRP, RSRQ [56], and expose it to third-party applications running on the MEC platform.

HOMonitor operates as follows: *firstly*, since RNIS only serves MEC internal applications, we design HOMonitor as a special MEC application to register the service with RNIS, which uses the standard Mp1 interface between "MEC Applications" and "MEC Services" [25], [26] to communicate with RNIS. *Then*, HOMonitor continuously uses *ue_id* (IMSI, unique identifier of a UE) as the index to obtain the latest RNI of the corresponding UE from RNIS. Meanwhile, HOMonitor builds and maintains a mapping table between *ue_id* and serving 5G gNB PCI (physical cell identifier). As shown in Fig. 14, this table will be refreshed with a time granularity of 1 ms and used to analyze UEs' HO information. When the table changes, for instance, the PCI corresponding to *ue_id 1335* changes from *135* to *27*, it will be regarded as a HO event and be submitted to Octopus's feedback module.

Monitoring subscription. If the HOMonitor actively monitors all UEs, the resource overhead will be significant and responsiveness will be compromised. Observing that most UEs are static or low-mobility, we design Octopus service subscription manager API, which allows a UE application to subscribe to Octopus only when necessary, so as to

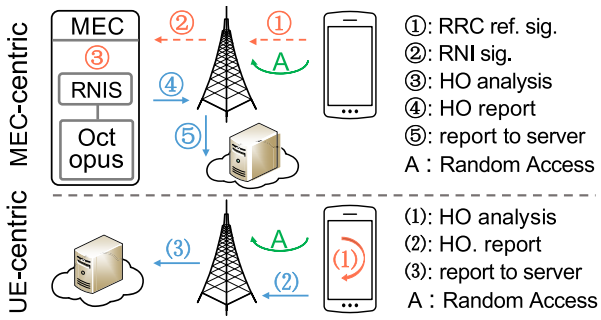

Fig. 15. Octopus's factor analysis.

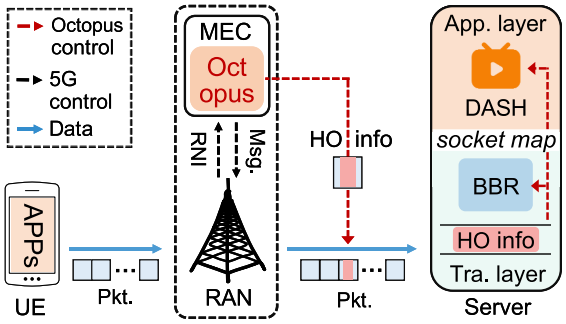

Fig. 16. Octopus's HO reporting mechanism.

TABLE IV
HO NOTIFICATION LATENCY OF Octopus

	MEC-centric				UE-centric		
breakd-own (ms)	①	②④	③	A	(1)	(2)	A
	1.05	0.01	0.53	17.2	17.14	1.05	17.2
entirety (ms)	**1.60** ①+②+③+④				**18.25** max {(1), A}+(2)		

TABLE V
COMPARISON OF SUBSCRIBE/NOTIFY METHODOLOGIES

	Octopus	**REST-based model**
Aware of MEC IP/hostname	○	●
In-band feedback	●	○
Extra MEC-server channel	○	●

reduce Octopus overhead. In particular, a UE application can establish an HTTP connection with the Octopus feedback module (*i.e.*, *HONotifier*) to initiate a subscription. Meanwhile, the HTTP request needs to carry *(i)* the UE's unique identifier (*i.e.*, *ue_id*), for Octopus to identify UE and detect HO; *(ii)* the TCP/UDP connection information (IP, port, *etc*) between itself and the application server, for Octopus to construct the HO reporting channel introduced later. Both of the above are easy to obtain for applications. We note that this subscription process is optional but not necessary, and Octopus can work without any change on UE.

Latency comparison: a back-of-envelope evaluation. Here we focus on downlink tasks that take up most of the Internet traffic, *i.e.*, to inform remote servers, and we will discuss the optimization of Octopus for uplink tasks in Sec. VII. Fig. 15 makes a comparison between the two approaches and breaks down their operating procedures. *For MEC-centric approach*, the UE's end-of-HO signaling is sent to the gNB (step ①), and Octopus over MEC monitors and analyzes the UE's HO state (steps ②, ③), and feeds back the HO information to the remote server (steps ④, ⑤). *In contrast, for UE-centric solution*, it continuously grabs and parses the UE's RRC signaling to determine the UE handover (step *(1)*). When HO is detected, it sends HO feedback to the remote server via gNB (steps *(2)*, *(3)*). Note that after the UE sends the end-of-HO signaling to the gNB, the UE must *randomly access* the new gNB (uplink synchronization) to resume data transmission (about 17.2 ms), which will affect the feedback delay. In particular, for feeding back to the remote server, the latency $T_{mec\_to\_server}$ = ①+②+③+④+⑤, whereas $T_{ue\_to\_server}$ = max{*(1)*, *A*}+*(2)*+*(3)*. We detail the latency overhead of each part in Table IV (omitting the transmission overhead from gNB to the server as it is the same for both approaches, and the results of steps ② and ④ are fluctuant). We can observe that to inform the remote server of a HO, the latency of Octopus is only $\frac{1}{10}$ compared with the UE-centric approach.

C. Handover Notification

For Octopus, HONotifier is responsible for informing the servers of a HO event. Fig. 13 shows the 3 sub-modules of HONotifier. When HONotifier receives the request from a UE (if monitoring subscription is applied) or detects any new UE association (otherwise), it first extracts the *ue_id* and then calls HOMonitor to monitor the corresponding UE's HO state. Once HOMonitor detects a HO event, it reports to HONotifier, which handles the notification process, using a middle-box or a *socket*-based cross-layer reporting channel, for server transport layer or application layer, according to subscription requirement.

1) Notifying the Server Transport Layer: A straightforward way to report the HO is through a dedicated tunnel between Octopus and the remote server, like the existing (REST-based) Subscribe/Notify model. But this solution has two obvious flaws: *(i)* It requires an additional TCP/UDP connection between the MEC and server, which not only induces resource overhead and latency, but also breaks the transparency of Octopus to the end hosts and violates the end-to-end design principle. *(ii)* Application servers must be aware of the IP address or hostnames of Octopus, which will require additional configuration in not only MEC but also application server. Such (configuration) considerations may dissuade application providers from Octopus.

Octopus's solution. We adopt an in-band feedback mechanism in HONotifier, which notifies the HO events by reusing the existing TCP connection between a local UE and the server, as shown in Fig. 16 (Table V summarizes the differences between two Subscribe/Notify schemes). Inspired by the middlebox scheme of cellular operators [57], [58], HONotifier first extracts the connection information (IP:port) from the UE application's subscription/association request to Octopus. It then generates a TCP packet *(based on raw socket)* [59], and enables one of the reserved bits in the TCP header (Fig. 17 (a)) to indicate the HO event. This special TCP packet is cached in HONotifier and will be injected into the network through the "MEC Data Plane" (the part of a MEC that carries user

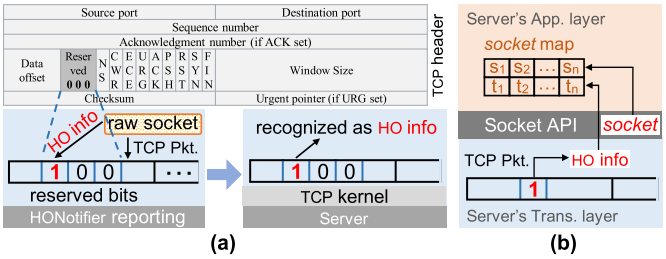

Fig. 17. (a) The structure and analysis of TCP HO signaling; (b) Server's application-layer reporting.

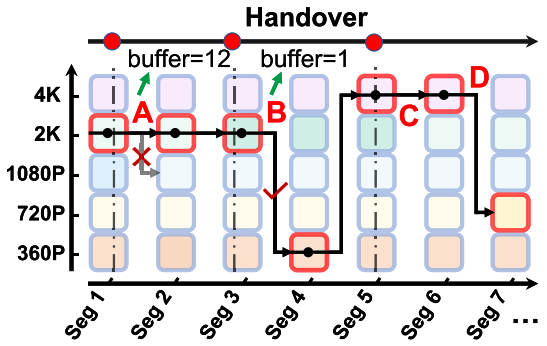

Fig. 18. DASH-H optimization strategy.

traffic between MEC and Internet [25]), upon a HO. Since the IP:port of this pre-generated packet is exactly the same as that of a normal packet, it will be routed to server to benefit its transport layer performance, like BBR.

It is noteworthy that: *(i)* We only re-utilize one of the reserved bits already in TCP header when generating the TCP packet carrying the HO information, which does not break TCP standard format or cause any incorrect checksum, as in previous works [20], [60]. *(ii)* As a result, the added packet will not interfere with the server's TCP state machine, and will not be dropped by Internet firewalls as we have validated in the operational 5G networks.

2) Notifying the Server Application Layer: Same as the transport layer, we are still committed to maintaining the transparency and reducing the network overhead of Octopus to MEC. So naively creating an HTTP connection with the server to report HO is not feasible.

Octopus's solution. We design a *socket*-based coordinated method for server application layer. In computer networks, an application (*e.g.*, HTTP, FTP flow) calls *socket* to establish a transport-layer flow and perform data transmission, *i.e.*, the application layer flow and the corresponding transport layer flow share the same *socket*, so they will be affected by the same HO. Therefore, we can store the HO information received by server transport layer with the *socket ID* as the index (Fig. 17 (b), similarly, we enable UDP-based application coordination by setting the reserved IP packet header flag bit). Then, the server application can obtain the corresponding UE's HO information according to the *socket ID*, which in consequence is used for its custom-designed HO adaptation, as shown in Sec. IV.

IV. Octopus Use Cases

It is noteworthy that Octopus is a general framework that can provide accessible 5G handover for various upper-layer applications. Here we develop two use cases to demonstrate how Octopus boosts network performance at the transport layer and application layer, respectively.

BBR-H. Here we show how Octopus can easily upgrade BBR towards a more efficient protocol, referred to as BBR-H. Specifically, upon server receiving the HO report from Octopus, BBR-H records the sequence number of the latest packet it sent. Then it uses this sequence number ($Seq._{HO}$) to judge whether the arriving ACK packets are delayed by the HO (*i.e.*, whether the acknowledgment numbers of these ACK packets are less than $Seq._{HO}$). If so, we neglect the impact of these delayed ACK, by modifying the calculation of $interval\_us_t$ as follows:

$$interval\_us'_t = \begin{cases} interval\_us_t, & Seq._{ACKed} > Seq._{HO} \\ RT_{prob}, & Seq._{ACKed} \leq Seq._{HO}, \end{cases} \quad (2)$$

where $Seq._{HO}$ is the sequence number of the packet right before the HO, and $Seq._{ACKed}$ is that of the packet being ACKed. RT_{prob} is the minimum RTT of the network link in the latest period of time (up to 10s) [30]. Using such adaptation, BBR-H prevents the overly conservative estimation of CWND. As $B_{tl}BW$ recovers, BBR-H increases the packet sending rate, which helps to converge to the new network bandwidth faster after a HO. Note that BBR-H can only take action when the UE resumes the data connection after HO and server receives the delayed ACK packets from UE, so even if the UE fails to handover, it will not cause side effects.

DASH-H. Similarly, we propose DASH-H to boost DASH performance over 5G. In detail, each time the client ABR algorithm requests to lower the bitrate of the next video segment, server-side DASH-H intervenes in the decision. It will query the time of the last HO event reported by Octopus and the instantaneous video buffer-level from DASH client resource request message, based on which it selects one of the following decisions (Fig. 18): *Type A*, if a HO (marked as ●) indeed happened since the last decision is made, and the video buffer is large enough (≥5s, enough to download the next video segment to keep playing, even 4K), DASH-H will prevent the ABR decision from lowering the bitrate (marked as ×) but hold on the current bitrate to respond the next video segment to client. *Type B*, if the buffer is insufficient after HO (*e.g.*, only 1s in this case due to HO failure), DASH-H will allow this decision of dropping bitrate to avoid video stalling (marked as ✓). Otherwise, if the HO does not impact the ABR algorithm's bitrate decision (*type C*), or there is no recent HO event, *i.e.*, the bitrate drop results from normal network bandwidth fluctuation (*type D*), DASH-H will approve the decision. All in all, DASH-H shields inaccurate ABR decisions caused by HO events.

V. Evaluation

A. Implementation

Octopus prototype. In a nutshell, we implement Octopus prototype (Fig. 19) in two steps. *Firstly*, we integrate two open-sourced platforms, LightEdge [61] and 5G-EmPOWER [62],

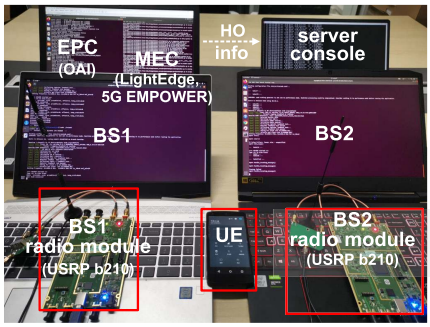

Fig. 19. Octopus prototype.

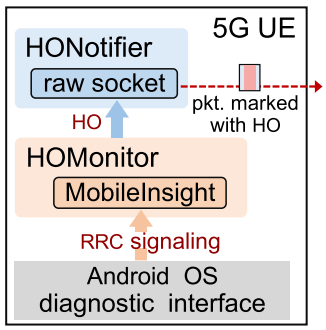

Fig. 20. Octopus approximate implementation (UE-centric).

to form the standard-compatible 5G-MEC framework for Octopus. The former is a MEC platform that is ETSI-compliant, for both 4G and 5G networks, and the latter is essentially a cellular RAN (based on srsRAN [63]) controller, which provides RAN state visibility. *Secondly*, we incorporate Octopus's two modules into the framework as follows: *(i)* HOMonitor is implemented by substantializing as a LightEdge MAF (MEC Application Function). It accesses to *lightedge-rniservice-manager microservice* (RNIS API, collaborating with 5G-EmPOWER) to grab *ue_id* and PCI, so as to intercept HO events. *(ii)* HONotifier is also deployed as a LightEdge MAF. On the one hand, it opens the service subscription interface using "flask" [64], a micro web framework written in Python that supports the common HTTP methods, including GET, POST, PUT, PATCH, and DELETE. On the other hand, HONotifier generates HO TCP/UDP packets using "scapy" [65], an interactive packet manipulation software, and then sends it to the server.

Currently, we are unable to deploy Octopus inside the operational 5G MEC, since we have no control over operational 5G. As shown in Fig. 20, we solve the problem by using an approximate deployment, *i.e.*, implementing Octopus's core function on a rooted 5G UE (UE-centric approach) in the real HSR 5G networks. It should be explained that the approximate deployment is based on MobileInsight beta-6.0.0 [66], which can replace MEC RNIS to monitor HO events via capturing RRC signaling from the UE diagnostic interface. In detail, *(i)* we re-develop MobileInsight to build HO monitoring module (*i.e.*, HOMonitor), which extracts HO information through the MobileInsight signaling analyzer, and then exposes the HO information to handover notification module (*i.e.*, HONotifier) in real-time by means of local shared memory. *(ii)* As for HONitifier, we obtain the header

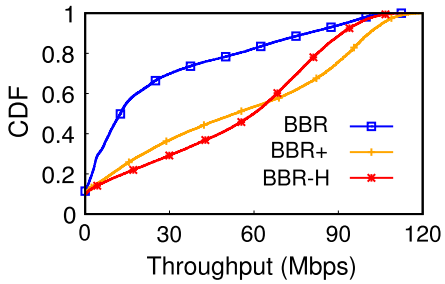

Fig. 21. Throughput improvement.

TABLE VI
TCP PERFORMANCE COMPARISON

Metrics	Throughput (Avg. ± Std. in Mbps)	RTT (25/50/75th per. in ms)
BBR	26.0 ± 29.9	57 / 78 / 171
BBR+	52.5 ± 38.0	190 / 264 / 497
BBR-H	52.2 ± 32.5	66 / 102 / 253
CUBIC	43.7 ± 36.4	74 / 170 / 397

information of the ACK packet sent by the local UE to the application server in the local UE OS kernel. Then we use the raw socket tool (Scapy [65]) to generate the ACK packet carrying the HO information, and send it to the application server via the local transmission interface. Essentially, there are no differences other than the deployment location where the ideal HONotifier is designed. We will discuss the performance difference between this approximate deployment and the ideal Octopus in Sec. V-F.

We first evaluate how much Octopus can help TCP and DASH (BBR-H and DASH-H, our use cases in Sec. IV) under the most critical HSR scenario, through extensive field tests. Then we demonstrate that Octopus works seamlessly under other scenarios with fewer HO events, including driving and walking.

B. TCP Performance Improvement

We focus on the state-of-the-art BBR with wide-scale commercial deployment. We set up a BBR session between the cloud server and 5G UE, and record the throughput and CWND. We compare three versions of BBR: the standard BBR [30], the recent adaptation for HSR called BBR+ [16], and the proposed BBR-H. For a fair comparison, we use the same experimental configurations for these algorithms, and each experiment is repeated for 3 round trips (1,044 Km).

Overall throughput. We plot the CDF of the measured throughput in Fig. 21, and more statistics in Table VI. We observe: *(i)* Both BBR-H and BBR+ increase average TCP throughput significantly over the original BBR, *i.e.*, by 2.0×. Moreover, they outperform CUBIC (which is less sensitive to HO than the vanilla BBR) by about 20%. *(ii)* The TCP throughput fluctuates in high dynamics, *i.e.*, the std. is comparable to the average, due to the impact of high mobility. The std. of BBR+ is 16.9% more than that of BBR-H, which indicates that BBR-H is more stable.

BBR-H optimally balances the throughput-delay. From Table VI, both BBR+ and BBR-H improve throughput at the

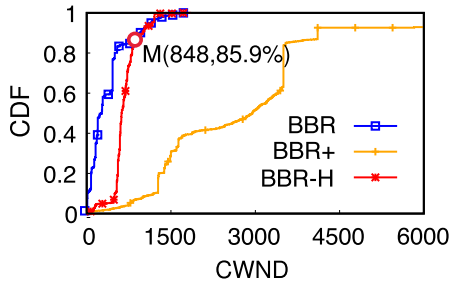

Fig. 22. CWND statistics.

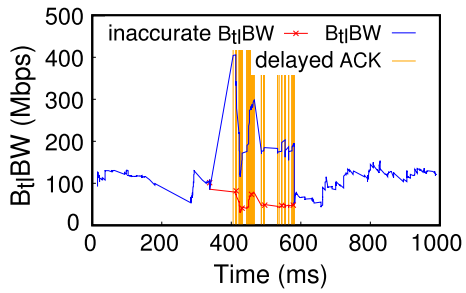

Fig. 23. $B_{tl}BW$ correction.

cost of extra delay. However, BBR-H sacrifices much less, *i.e.*, the RTT is 1.16×, 1.31×, and 1.48× at RTT 25/50/75th percentage than BBR, respectively, whereas for BBR+ RTT is 3.34×, 3.40×, 2.87× of BBR, respectively. Compared with CUBIC, BBR-H even reduces RTT by 40% on average, while simultaneously achieving 20% higher throughput. To understand this, we conduct a microscopic analysis as follows.

BBR-H increases CWND only when necessary. We compare the CDF of CWND under each algorithm in Fig. 22. The first observation is that BBR-H successfully avoids the mistaken CWND reduction, *e.g.*, 60% BBR CWND is lower than 400 since it is deceived by the frequent HO events, while 90% CWND of BBR-H is higher than 400. One interesting detail is that the CWND value of BBR and BBR-H intersects and coincides after point M, with the coordinate of (848, 85.9%), *i.e.*, the high-CWND zone. The result indicates that about 14.1% congestion control strategy of BBR-H and BBR allocates the same CWND value. Given that high CWND usually occurs when the link is stable, BBR-H only corrects the inaccurate bandwidth probing of BBR due to HO events, while not disturbing BBR's normal decisions.

Essentially, BBR-H improves TCP throughput by correcting BBR's $B_{tl}BW$ deviation. Here we showcase such a process using a 1000 ms trace in Fig. 23. We observe that when the delayed ACK packets affected by HO arrive, the $B_{tl}BW$ is mistakenly lowered, which lasts about 180 ms with a minimum of only 30 Mbps. This is caused by the inaccurate timing of the $interval\_us_t$ detector under frequent HO events. In contrast, BBR-H can acquire the HO information and adjust the $interval\_us_t$ accordingly, following Eq. 2. Consequently, the adjusted $B_{tl}BW$ helps BBR to maintain the original CWND, and achieve higher throughput in Fig. 21.

On the other hand, BBR+ expands CWND to very large values, while not bringing higher throughput than BBR-H. The reason is that BBR+'s adjustment is blindly aggressive without knowing HO information. In particular, BBR+ changes CWND gain sequence ($cycle\_pacing\_gain$) from [5/4, 3/5, 1, 1, 1, 1, 1, 1] to [3/2, 1/2]) and RT_{prob} by adding a compensation term [16]. As a result, BBR+ overestimates CWND and prolongs the queuing delay, and finally causes excessive RTT. Meanwhile, the modification of hyperparameters by BBR+ also leads to its limited application scenarios. We find that the RTT of BBR and BBR-H is almost the same in static, while the RTT of BBR+ is 3.6× that of them, which brings too many flying bytes to the network link. To sum up, ***Octopus**'s ability to detect and react to HO is the fundamental reason for achieving the optimal throughput-delay balance.*

TABLE VII
PERFORMANCE OF DIFFERENT ABR ALGORITHMS

Metrics	Avg. Bitrate (Mbps)	Oscillation (Mbps)	Stalling Rate (%)
Thro. / Thro.-H	14.5 / 20.4	5.6 / 1.2	3.9 / 2.7
Bola / Bola-H	13.7 / 20.5	5.6 / 1.9	0.8 / 0.1
Dyn. / Dyn.-H	16.1 / 21.2	4.1 / 1.4	2.3 / 1.6

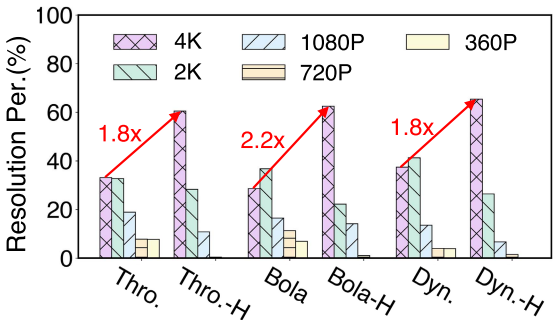

Fig. 24. Resolution (bitrate) distribution.

C. VoD Performance Improvement

We compare DASH-H performance against three de-facto ABR algorithms: throughput-based (Throughput), buffer-based (Bola), and hybrid (Dynamic) in the widely-used *dash.js* framework [31], [42], [67]. We stream the same video following the setup in Sec. II-A across 8 round trips along the HSR. We summarize the performance statistics in Table VII.

DASH-H guarantees high-resolution video playback. DASH-H improves the average bitrate significantly, *i.e.*, by 1.4×, 1.5×, and 1.3× for three ABR algorithms, respectively. The corresponding 4K playback percentage for Throughput-H, Bola-H, and Dynamic-H is 60.45%, 62.46%, and 65.33% respectively, which are approximately 2× over the corresponding baseline methods (Fig. 24). The result demonstrates that DASH-H can indeed push the limit of the network, boosting the vanilla algorithm to an ideal video transport algorithm that can support 4K video playback 64.5% of the time over 5G (Fig. 9). Note that there remains a small percentage of low-resolution (<1080P) video playback with DASH-H, *e.g.*, 0.49%, 1.13%, and 1.53% for Throughput-H, Bola-H, and Dynamic-H respectively. We attribute this to the 12.7% of physical layer handover failures (shown in Table II), during which TCP suffers from low throughput and even connection reset, and DASH-H chooses to reduce the video bitrate to avoid playback stalling.

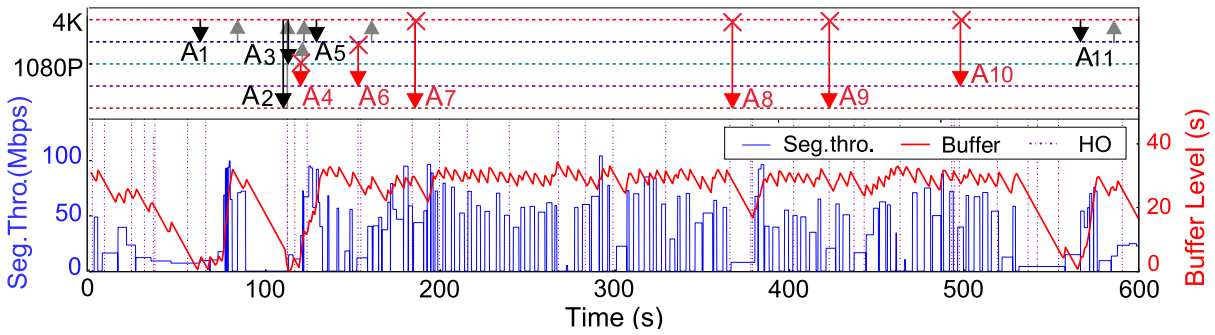

Fig. 25. Showcase of DASH-H's bitrate decisions.

Bitrate oscillation. We now examine the video smoothness of each algorithm, using the bitrate oscillation ρ – the average bitrate difference between the adjacent segments:

$$\rho = \frac{D * \sum_{i=2}^{N} |B(i) - B(i-1)|}{T}, \tag{3}$$

where D is the duration of a video segment, 1.6 seconds in our system. $B(i)$ is the bitrate of each playback segment, and T is the total playback time. From Table VII, we observe that DASH-H reduces the video bitrate oscillation significantly, by 78.6%, 66.1%, 65.9%, respectively, as it eliminates inaccurate bitrate fluctuations caused by HO events.

Stalling rate. We define the stalling rate as the fraction of stall time over the whole playback duration. Our major finding (Table VII) is that Octopus helps all three ABR algorithms, despite their heterogeneity, *i.e.*, the stalling rate is reduced by 30.8%, 87.5%, 30.4%, respectively. Among them, Bola-H achieves the most remarkable stalling reduction, while still maintaining a competitive bitrate. The result implies that Octopus realizes the full potential when integrated with the buffer-based ABR algorithm.

A showcase: DASH-H decision optimization. We then showcase how DASH-H corrects inaccurate decisions effectively. The upper part of Fig. 25 records DASH's bitrate adjustment, and the lower part shows the segment throughput and buffer-level. We can find that there are 11 DASH actions to reduce the video bitrate, marked as ↓ and ↓. Besides, ↑ means to improve the bitrate. We divide the above bitrate reduction handlers into two categories: *(i)* For {A_1 - A_3, A_5, and A_{11}}, there is a significant setback of low segment throughput or insufficient buffer. ABR algorithm (Dynamic) naturally adjusts its resolution choice to prevent stalling. *(ii)* The inaccurate decisions of ABR occur at point {A_4, A_6 - A_{10}}, where multiple short-term throughput or buffer-level drops are caused by HO. However, the buffer is still sufficient (even beyond 10s). Therefore, Dynamic-H executes its internal mechanism to correct ABR decisions (marked as ×) to keep high bitrates without the risk of video stalling.

D. Octopus Benefits Consistently in 4G Network

We evaluate Octopus in 4G under the same setup with Sec. V-B and Sec. V-C. Despite lower absolute TCP throughput and Dash video bitrate, we find that *(i)* Octopus improves TCP throughput by 1.2×, from 19.0±20.5 Mbps to 22.8±27.1 Mbps. *(ii)* The change of ≥2K video playback ratio is 48.7% → 66.0%, with average bitrate from 9.5 to 10.4 Mbps. All in all, as a general MEC service framework, Octopus not only benefits 5G but also the more widely used 4G nowadays.

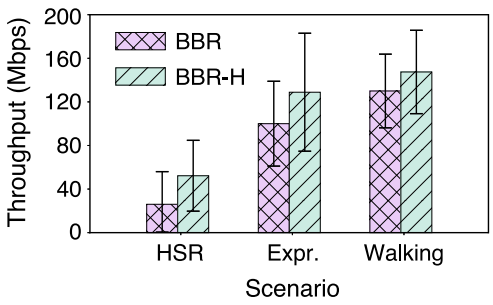

Fig. 26. BBR vs. BBR-H under different scenes.

E. Octopus's Applicability Beyond HSR Scenarios

We now examine how Octopus helps TCP and video streaming under driving and walking scenarios. We carry out experiments on a car traveling on city highways, at the speed of 60-120 Km/h. The route is 270 Km over 3 hours, during which we observe 765 HO events. Similarly, for the walking scenario, we conduct the same experiments while walking on a university campus at the speed of 4-5 Km/h. The total route is 18 Km, during which 75 HO events happen, as gNBs in the campus are denser.

We present the resulting TCP performance in Fig. 26, from which we have two observations: *(i)* As expected, the average throughput of the vanilla BBR is relatively higher: 100.8 Mbps (driving) and 129.8 Mbps (walking) in comparison to the HSR case (26.0 Mbps), as handover becomes much less frequent. *(ii)* Nevertheless, Octopus still achieves 1.3× and 1.1× throughput gain for the driving and walking scenarios, respectively. This validates the general advantage of Octopus: it improves TCP performance even under low mobility, as long as HO events are involved. For DASH VoD, Fig. 27. shows that Octopus enhances video quality across all scenarios, *e.g.*, the changes of 4K video playback ratio are 37.4% → 65.3%, 51.4% → 85.7%, and 73.5% → 95.4% for the three scenarios, respectively. We observe that Octopus can bring about 1.3× gain in terms of 4K playtime ratio, even for the slowest walking scenario.

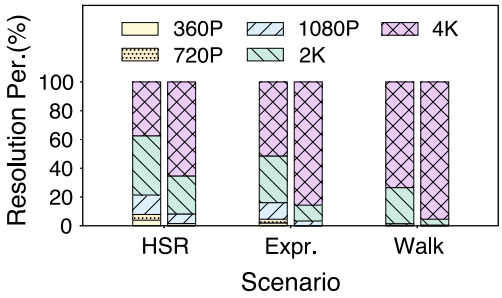

Fig. 27. DASH (left) vs. DASH-H (right) under different scenes.

TABLE VIII
SLIGHT IMPACT OF IMPL. APPROXIMATION

Metrics	4K (%)	Avg. Bitrate (Mbps)	Oscillation (Mbps)	Stalling Rate (%)
Approx.	65.01	20.81	1.37	1.80
Ideal	65.86	20.97	1.27	2.00

F. Impact of Implementation Approximation

As shown in Sec. III-B, the HO detection and reporting deployed on the UE-side that we used in the field experiments are slightly different from Octopus, and eventually lead to different HO reporting latency. Such difference also propagates to the application layer. Here we repeat the DASH-H experiments using the HSR DASH traces, while intentionally advancing 16.65 ms (the ideal system delay difference comes from Table IV) to report HO to the server application layer, so as to simulate Octopus's ideal implementation. From the results shown in Table VIII, our approximate deployment testbed and the ideal Octopus have very close gains to DASH, in terms of all four comparison metrics. For instance, the differences are only 0.85% and 0.20% for 4K playback ratio and the stalling rate, respectively. We analyze the traces and find the underlying reason: DASH usually decreases the bitrate after the negative effect of HO persists for a second-level period, which is much longer than 16.65 ms.

In summary, the result indicates that Octopus could be an efficient mobile network service compatible with the 5G MEC framework. Looking into the future, more latency-critical applications, like augmented reality [3], [68] or real-time video analytics [69], [70] will run on the edge. These applications may leave more improvement space for Octopus, as it can report HO to the edge server 16.65 ms in advance (Table IV).

VI. RELATED WORK

5G measurement and optimization has accompanied its recent roll-out. Xu et al. [12] measured the sub-6 GHz 5G networks, while Narayanan et al. [14] focused on the 5G mmWave band and explored using environmental information to predict network performance. Zhang et al. [71] measured the path loss of 5G mmWave and optimized the existing 3GPP channel models. Li et al. [72] improved cellular reliability from a software perspective. Unlike the prior research, Octopus focuses on the 5G HO problems under high mobility, discovering the alarming impacts on upper layers and developing a solution framework accordingly.

Ultra-high mobility management. Previous work measured the 4G performance in high mobility. Li et al. [15] revealed the poor TCP adaptation to high mobility, and proposed a multi-path solution. Wang et al. [16] further analyzed the root cause of TCP's abnormality in 4G HSR scenario, and tailored TCP BBR. In [18], Li et al. found that the existing handover strategy caused too many HO failures, and proposed a movement-based scheme. Another recent work [17] advocates to re-design the cellular control plane for reliable and low-latency cellular access. While drawing inspiration from them, we go beyond to analyze the operational 5G in HSR scenario at scale, and introduce a general-purpose MEC service framework that enhances upper-layer performance.

5G MEC. 3GPP and ETSI are standardizing 5G MEC [49], [50], [56] as a key architectural concept to meet the demanding requirements of 5G applications [73], [74]. Recent research starts to explore the advantages of integrating MEC into 5G [75], [76], [77], but mostly limited to analytical or simulation study for feasibility or concept verification. In this work, we build and implement Octopus using a standard 5G MEC framework, and validate its practical gain in the wild.

TCP and VoD optimization. As a persistent problem along with the Internet's evolution, *(i)* TCP evolves from traditionally newReno [78], Cubic [48] to BBR [30] and PCC [79], and has also tailed for cellular networks [20], [80], [81], [82], or for better delivering real-time video [83], [84], [85]. *(ii)* Video streaming ABR algorithms are traditionally designed based on instantaneous throughput and buffer-level [67], [86], or further enhanced by physical layer information [22]. Emerging algorithms adopt machine learning for QoE optimization [87], [88], [89]. In contrast, we do not design any new TCP or VoD ABR algorithms. Rather, we propose a generic MEC service framework that can be utilized by any algorithms.

VII. DISCUSSION AND FUTURE WORK

Handover information exposure. Technically, mobile operators can disclose user mobility information (Ho Notify) through MME/AMF. However, in practice, operators (at least in the 4G phase) refused to provide any RAN or core network scheduling information to application providers or mobile users, due to concerns such as the security of the cellular network stack. Until very recently, the latest 3GPP 5G standard [23] proposes NEF (Network Exposure Function) to open network capabilities, resource allocation, and other information to third-party applications, but it is still not mature yet, *i.e.*, in the stage of protocol formulation. In addition, considering the massive concurrency, HO Notify service based on MME/AMF will bring huge overhead to the already overwhelmed core network. On the other hand, Octopus represents a pioneering implementation of 5G NEF, designed to be deployed on distributed MEC servers to provide mobility support for massive UEs. We summarize the above comparisons in Table IX.

Octopus along with 5G evolution. In this paper, we focus on the well-covered Sub-6 GHz 5G networks, different from the more environmentally-sensitive mmWave-band 5G [13], [14]. In addition, we focus on the impact of extreme mobility on 5G, so we intentionally choose the HSR with good coverage (our UEs access 5G for more than 90% of time) and filter out the few 4G-only access cases. We believe that Octopus will

TABLE IX
IMPACT OF HO NOTIFY DEPLOYMENT LOCATION

	Octopus	HO Notify by MME/AMF
Implementation location	MEC	5G core network
Topology structure	Distributed	Centralized
Extra overhead on core network	No	Yes
Compatible with 5G NEF	Yes	No

also help under other handover types, such as 5G↔4G, which is left for future exploration.

Octopus is evaluated under the currently 5G NSA architecture. In the near future, 5G will move forward to the more advanced Standalone (SA) architecture, in which it owns independent EPC and thus will reduce 5G HO duration [13], [23]. However, the increasing 5G gNBs density will bring more frequent HOs, especially for mmWave 5G. Moreover, TCP, VoD, and other applications still suffer from passive adaptation to the frequent 5G HO events. Hence we believe Octopus will still play a significant role in the future 5G networks.

Octopus for optimizing uplink performance. In this paper, Octopus is designed for downlink traffic (taking the dominating 93.3% fraction of the whole Internet traffic [90]), where HO adaptation is executed on back-end servers when sending data to UEs. Octopus can also help the opposite direction uplink traffic, once the UE side can be updated to receive and respond to the HO notification from Octopus's MEC side, which will be more practical after MEC-UE signaling is standardized.

Extending Octopus across multiple MECs. A continuously moving UE may leave the coverage of multiple gNBs served by one MEC, and thus out of the service range of Octopus on that MEC. However, the problem can be solved by incorporating Octopus with MEC's own mobility support [24], [25]. In particular, Octopus, as a MEC service can be deployed on many MEC servers distributively, *i.e.*, one Octopus instance for each MEC server. To ensure that MEC applications are not affected by UE mobility, MEC constructs the application redirection between MEC servers to provide application mobility support and V2X support. When the UE continues to move and is about to leave the range of current MEC server, the current (source) MEC server will prepare to migrate UE-related application instances to the adjacent (target) MEC server to continue serving the UE. For Octopus, MEC orchestrator will enable Mp3 interface on source and target MEC servers for control communication interaction, associate Octopus state from source MEC server to target MEC server, and synchronize UE's context. Thus the UE, once out of source Octopus's range, can be served by the target Octopus instance on the neighboring MEC server, and the source Octopus instance will stop the service. In addition, in 5G SA, MEC will be deployed in RAN centralized unit (CU) commonly serving an 80 Km radius area [50]. Hence even in extreme mobility (∼300 Km/h), the mobile UE will experience very infrequent MEC switches, *i.e.*, less than 4 switches per hour, which lead to acceptable overhead.

VIII. CONCLUSION

In this work, we find 5G's gain is severely undermined under high mobility scenarios, *e.g.*, HSR or driving, and uncover the reason: the complex interaction among 5G handover events and applications' sluggish reaction. We thus propose Octopus, which harnesses 5G edge to enable accessible mobility management. More importantly, Octopus is standard-compatible and can be integrated into the 5G MEC framework as a universal service. We believe Octopus hints at the huge potential of improving 5G by exploiting its just-begun openness, *e.g.*, MEC, open RAN, capacity exposure, *etc*.

ACKNOWLEDGMENT

The authors appreciate the insightful feedback from the anonymous shepherd and reviewers who helped improve this work.

REFERENCES

[1] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.
[2] A. Gohil, H. Modi, and S. K. Patel, "5G technology of mobile communication: A survey," in *Proc. Int. Conf. Intell. Syst. Signal Process. (ISSP)*, Mar. 2013, pp. 288–292.
[3] A. Yaqoob, T. Bi, and G.-M. Muntean, "A survey on adaptive 360° video streaming: Solutions, challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2801–2838, 4th Quart., 2020.
[4] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim, "GROOT: A real-time streaming system of high-fidelity volumetric videos," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2020, p. 57.
[5] B. Han, Y. Liu, and F. Qian, "ViVo: Visibility-aware mobile volumetric video streaming," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, Apr. 2020, p. 11.
[6] H. Ning et al., "A survey on metaverse: The state-of-the-art, technologies, applications, and challenges," 2021, *arXiv:2111.09673*.
[7] M. Condoluci, M. A. Lema, T. Mahmoodi, and M. Dohler, "5G IoT industry verticals and network requirements," in *Powering the Internet of Things With 5G Networks*. Hershey, PA, USA: IGI Global, 2018, pp. 148–175.
[8] S. A. A. Hakeem, A. A. Hady, and H. Kim, "5G-V2X: Standardization, architecture, use cases, network-slicing, and edge-computing," *Wireless Netw.*, vol. 26, no. 8, pp. 6015–6041, Nov. 2020.
[9] Wikipedia. (2022). *High-Speed Rail*. [Online]. Available: https://en.wikipedia.org/wiki/High-speed_rail
[10] (2022). *China HSR Passenger Flow Statistics*. [Online]. Available: https://en.wikipedia.org/wiki/High-speed_rail_in_China
[11] (2020). *China Flight Passenger Flow Statistics*. [Online]. Available: https://www.mot.gov.cn/tongjishuju/minhang/202106/P020210616371268778138.pdf
[12] D. Xu et al., "Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption," in *Proc. SIGCOMM*, 2020, pp. 479–494.
[13] A. Narayanan et al., "A variegated look at 5G in the wild: Performance, power, and QoE implications," in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 610–625.
[14] A. Narayanan et al., "A first look at commercial 5G performance on smartphones," in *Proc. Web Conf.*, 2020, pp. 894–905.
[15] L. Li et al., "A longitudinal measurement study of TCP performance and behavior in 3G/4G networks over high speed rails," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2195–2208, Aug. 2017.
[16] J. Wang et al., "An active-passive measurement study of TCP performance over LTE on high-speed rails," in *Proc. MobiCom*, 2019, pp. 18:1–18:16.
[17] M. Ahmad et al., "A low latency and consistent cellular control plane," in *Proc. SIGCOMM*, 2020, pp. 648–661.
[18] Y. Li, Q. Li, Z. Zhang, G. Baig, L. Qiu, and S. Lu, "Beyond 5G: Reliable extreme mobility management," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, Jul. 2020, pp. 344–358.
[19] Y. Li et al., "Experience: A five-year retrospective of mobileinsight," in *Proc. MobiCom*, 2021, pp. 28–41.
[20] X. Xie, X. Zhang, and S. Zhu, "Accelerating mobile web loading using cellular link information," in *Proc. 15th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2017, pp. 427–439.

[21] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang, "Mobileinsight: Extracting and analyzing cellular network information on smartphones," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2016, pp. 202–215.

[22] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "PiStream: Physical layer informed adaptive video streaming over LTE," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2015, pp. 413–425.

[23] (2020). *3GPP R16*. [Online]. Available: https://www.3gpp.org/release-16

[24] ETSI (2019). *MEC Edge Platform Application Enablement*. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/011/02.01.01_60/gs_mec011v020101p.pdf

[25] ETSI (2019). *MEC Framework and Reference Architecture*. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/mec/001_099/003/02.01.01_60/gs_mec003v020101p.pdf

[26] ETSI (2019). *Radio Network Information API*. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/012/02.01.01_60/gs_mec012v020101p.pdf

[27] RcrWirelessNews. (2021). *A Closer Look at Verizon's 5G MEC Strategy*. [Online]. Available: https://www.rcrwireless.com/20210108/5g/a-closer-look-at-verizons-5g-mec-strategy

[28] S. Newsroom. (2020). *Samsung and KT Complete Korea's First 5G SA and NSA Common Core Network Deployment*. [Online]. Available: https://bit.ly/2TKwXTw

[29] Channelasia. (2021). *Singtel Steps Up 5G Standalone Access Across Singapore*. [Online]. Available: https://www.channelasia.tech/article/687963/singtel-steps-up-5g-standalone-access-across-singapore/

[30] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Netw. Congestion*, vol. 14, pp. 20–53, Dec. 2016.

[31] MPEG. (2017). *Dynamic Adaptive Streaming Over HTTP*. [Online]. Available: https://mpeg.chiariglione.org/standards/mpeg-dash

[32] H. Deng, C. Peng, A. Fida, J. Meng, and Y. C. Hu, "Mobility support in cellular networks: A measurement study on its configurations and implications," in *Proc. Internet Meas. Conf.*, Oct. 2018, pp. 147–160.

[33] R. Merz, D. Wenger, D. Scanferla, and S. Mauron, "Performance of LTE in a high-velocity environment: A measurement study," in *Proc. 4th Workshop All Things Cellular, Oper., Appl., Challenges*, 2014, pp. 47–52.

[34] Y. Li, J. Xu, C. Peng, and S. Lu, "A first look at unstable mobility management in cellular networks," in *Proc. 17th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2016, pp. 15–20.

[35] 3GPP. (2019). *TS37.863: R15: E-Utra (Evolved Universal Terrestrial Radio Access)—NR Dual Connectivity (EN-DC)*. [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/37_series/37.863-02-01/

[36] (2020). *Airscreen: A Wireless Network Testing and Analysis Software*. [Online]. Available: https://www.qtrun.com/cn/?page_id=60

[37] (2019). *IPerf 3.7*. [Online]. Available: https://github.com/esnet/iperf/releases/tag/3.7

[38] (2019). *BBR Open Source Distribution*. [Online]. Available: https://github.com/google/bbr

[39] (2019). *Wireshark*. [Online]. Available: https://www.wireshark.org/

[40] (2021). *Nginx Instance Manager*. [Online]. Available: https://www.nginx.com/

[41] (2021). *Dash.js*. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js

[42] Wikipedia. (2018). *The ABR Logic of Dash.js*. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js/wiki/ABR-Logic/

[43] (2019). *Ffmpeg*. [Online]. Available: https://www.ffmpeg.org/

[44] (2020). *Bento4 MP4 and Dash Class Library, SDK and Tools*. [Online]. Available: https://www.bento4.com/

[45] 3GPP. (2020). *3GPP R15*. [Online]. Available: https://www.3gpp.org/release-15

[46] 3GPP. (2019). *TS36.331: Radio Resource Control (RRC)*. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/36_series/36.331

[47] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling bufferbloat in 3G/4G networks," in *Proc. ACM Conf. Internet Meas. Conf.*, 2012, pp. 329–342.

[48] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[49] (2018). *3GPP Enables MEC Over a 5G Core*. [Online]. Available: https://www.3gpp.org/news-events/partners-news/1969-mec

[50] S. Kekki et al., "MEC in 5G networks," *ETSI White Paper*, vol. 28, pp. 1–28, Jun. 2018.

[51] (2019). *3GPP Specification Numbering*. [Online]. Available: https://www.3gpp.org/specifications/specification-numbering

[52] E. G. Observatory. (2021). *Number of 5G Base Stations*. [Online]. Available: https://5gobservatory.eu/wp-content/uploads/2021/11/5G-Obs-PhaseIII_Quarterly-report-13_final-version-11112021.pdf

[53] (2019). *XCAL Mobile*. [Online]. Available: http://accuver.com/acv_products/xcal-mobile/

[54] Z. Tan, J. Zhao, Y. Li, Y. Xu, and S. Lu, "Device-based LTE latency reduction at the application layer," in *Proc. NSDI*, 2021, pp. 471–486.

[55] J. Gedeon, F. Brandherm, R. Egert, T. Grube, and M. Mühlhäuser, "What the fog? Edge computing revisited: Promises, applications and future challenges," *IEEE Access*, vol. 7, pp. 152847–152878, 2019.

[56] A. Reznik et al., "Cloud RAN and MEC: A perfect pairing," *ETSI White Paper*, vol. 23, pp. 1–24, Dec. 2018.

[57] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 374–385.

[58] F. Le, E. Nahum, V. Pappas, M. Touma, and D. Verma, "Experiences deploying a transparent split TCP middlebox and the implications for NFV," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Function Virtualization*, Aug. 2015, pp. 31–36.

[59] OpenSource. (2015). *Raw Socket*. [Online]. Available: https://www.opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/

[60] K. He et al., "AC/DC TCP: Virtual congestion control enforcement for datacenter networks," in *Proc. SIGCOMM*, 2016, pp. 244–257.

[61] (2021). *Lightedge*. [Online]. Available: https://lightedge.io/

[62] (2021). *5G-Empower: An O-Ran Compliant Near-RT Ran Intelligent Controller for HET Rans*. [Online]. Available: http://5g-empower.io/

[63] (2021). *SRSRAN*. [Online]. Available: https://www.srslte.com/

[64] (2021). *Flask*. [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/

[65] (2021). *Scapy*. [Online]. Available: https://scapy.net/

[66] (2020). *Mobileinsight 6.0 Beta for 5G*. [Online]. Available: http://www.mobileinsight.net/news-6.0.html

[67] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1698–1711, Aug. 2020.

[68] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, pp. 1–16, Jun. 2015.

[69] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-aware video analytics on edge computing platform," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1–13.

[70] C. Pritchard, Y. Beheshti, and M. Sepahi, "Mobile edge computing: Architecture, use-cases, applications," HAL Open Sci., Lyon, France, Tech. Rep. hal-02612631, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02612631 and https://hal.archives-ouvertes.fr/

[71] P. Zhang, B. Yang, C. Yi, H. Wang, and X. You, "Measurement-based 5G millimeter-wave propagation characterization in vegetated suburban macrocell environments," *IEEE Trans. Antennas Propag.*, vol. 68, no. 7, pp. 5556–5567, Jul. 2020.

[72] Y. Li et al., "A nationwide study on cellular reliability: Measurement, analysis, and enhancements," in *Proc. SIGCOMM*, 2021, pp. 597–609.

[73] Q. Yanli, Z. Yiqing, L. Ling, T. Lin, and S. Jinglin, "MEC coordinated future 5G mobile wireless networks," *J. Comput. Res. Develop.*, vol. 55, no. 3, p. 478, 2018.

[74] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Dec. 2017.

[75] X. Hou, Z. Ren, K. Yang, C. Chen, H. Zhang, and Y. Xiao, "IIoT-MEC: A novel mobile edge computing framework for 5G-enabled IIoT," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–7.

[76] X. Chen, Z. Liu, Y. Chen, and Z. Li, "Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks," *IEEE Access*, vol. 7, pp. 184172–184182, 2019.

[77] Q.-V. Pham et al., "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[78] S. Floyd, T. Henderson, and A. Gurtov, *The Newreno Modification to TCP's Fast Recovery Algorithm*, document RFC3782, 2004.

[79] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. NSDI*, 2015, pp. 395–408.

[80] 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. 14th Int. Conf. Emerg. Netw. EXperiments Technol.*, Dec. 2018, pp. 307–319.

[81] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 509–522.
[82] Y. Xie, F. Yi, and K. Jamieson, "PBE-CC: Congestion control via endpoint-centric, physical-layer bandwidth measurements," in *Proc. SIGCOMM*, 2020, pp. 451–464.
[83] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Congestion control for web real-time communication," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2629–2642, Oct. 2017.
[84] A. Zhou, "Learning to coordinate video codec with transport protocol for mobile video telephony," in *Proc. MobiCom*, 2019, pp. 29:1–29:16.
[85] H. Zhang et al., "OnRL: Improving mobile video telephony via online reinforcement learning," in *Proc. MobiCom*, 2020, pp. 29:1–29:14.
[86] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: Improving bitrate adaptation in the DASH reference player," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 15, no. 2s, pp. 1–29, 2019.
[87] V. Martín, J. Cabrera, and N. García, "Design, optimization and evaluation of a Q-learning HTTP adaptive streaming client," *IEEE Trans. Consum. Electron.*, vol. 62, no. 4, pp. 380–388, Nov. 2016.
[88] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 197–210.
[89] Z. Akhtar et al., "Oboe: Auto-tuning video ABR algorithms to network conditions," in *Proc. SIGCOMM*, 2018, pp. 44–58.
[90] (2021). *Network Traffic Patterns*. [Online]. Available: https://corporate.comcast.com/press/releases/comcast-2020-network-performance-data

Congkai An received the B.E. degree from Yanshan University in 2019. He is currently pursuing the Ph.D. degree in computer science and technology with the Beijing University of Posts and Telecommunications. His research interest lies in 4G/5G cellular networks, network congestion control, and video streaming transport optimization.

Anfu Zhou received B.S. degree from the Renmin University of China and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy Sciences, in 2012. He is currently a Professor with the School of Computer Science, Beijing University of Posts and Telecommunications. His research interest lies in mobile computing, wireless networking, and the IoT systems.

Jialiang Pei received the B.E. degree from the Beijing University of Post and Telecommunications, Beijing, China, in 2020, where he is currently pursuing the master's degree in computer science and technology. His research focuses on adaptive video streaming bitrate optimization based on DASH.

Xi Liu received the B.E. degree from the Beijing University of Post and Telecommunications, Beijing, China, in 2020, where he is currently pursuing the master's degree in computer science and technology. His research focuses on network congestion control.

Dongzhu Xu (Student Member, IEEE) received the B.E. degree from Qingdao University in 2018. He is currently pursuing the Ph.D. degree in computer science and technology with the Beijing University of Posts and Telecommunications. His research interest lies in 4G/5G cellular networks, mobile edge computing, mmWave networking systems, and network measurement.

Liang Liu received the B.S. degree from the Department of Computer Science and Technology, South China University of Technology, Guangzhou, China, in 2004, and the Ph.D. degree from the Department of Computer, Beijing University of Posts and Telecommunications, Beijing, China, in 2009. He was a Visiting Ph.D. Student at the Networking and Information Systems Laboratory, Texas A&M University, College Station, TX, USA, from 2007 to 2008. He is currently a Professor with the Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia and the Dean of the School of Computer Science, Beijing University of Posts and Telecommunications. He has published over 100 articles. His current research interests include the Internet of Things and intelligent sensing technologies.

Huadong Ma (Fellow, IEEE) received the B.S. degree in mathematics from Henan Normal University, Xinxiang, China, in 1984, the M.S. degree in computer science from the Shenyang Institute of Computing Technology, Chinese Academy of Science, Beijing, China, in 1990, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, in 1995. From 1999 to 2000, he held a Visiting Position at the University of Michigan, Ann Arbor, MI, USA. He is currently a Professor with the Beijing University of Posts and Telecommunications, China. He has published more than 300 papers in journals (such as ACM/IEEE Transactions) or conferences (such as ACM SIGCOMM, ACM MobiCom, and IEEE INFOCOM) and five books. His current research interests include the Internet of Things and sensor networks and multimedia computing. He received the Natural Science Award of the Ministry of Education, China, in 2017. He received the 2019 Prize Paper Award of IEEE TRANSACTIONS ON MULTIMEDIA, the 2018 Best Paper Award from IEEE MULTIMEDIA, the Best Paper Award in IEEE ICPADS 2010, and the Best Student Paper Award in IEEE ICME 2016 for his coauthored articles. He received the National Funds for Distinguished Young Scientists in 2009. He is an Editorial Board Member of the IEEE TRANSACTIONS ON MULTIMEDIA, IEEE INTERNET OF THINGS JOURNAL, and *ACM Transactions on Internet of Things*. He serves as the Chair of ACM SIGMOBILE China.