Supporting Mobile VR in LTE Networks: How Close Are We?

ZHAOWEI TAN, University of California, Los Angeles, USA
YUANJIE LI, University of California, Los Angeles, USA
QIANRU LI, University of California, Los Angeles, USA
ZHEHUI ZHANG, University of California, Los Angeles, USA
ZHEHAN LI, University of California, Los Angeles, USA
SONGWU LU, University of California, Los Angeles, USA

Mobile virtual reality (VR) headsets (e.g., Google Cardboard and Samsung Gear VR) seek to offer "anytime, anywhere" panorama, immerse 3D experiences for users. In this work, we study the viability of supporting mobile VR over operational 4G LTE networks, where the device provides pose information to the edge servers to offload graphical processing. We find that, contrary to common perceptions, wireless bandwidth is not the latency bottleneck for medium-quality VR. Instead, the signaling operations, which facilitate wireless data delivery, constitute a bulk portion of the latency. We report findings that challenge five common beliefs on VR network latency in LTE under both static and mobile scenarios, and quantify their impact. We design `LTE-VR`, a client-side solution to medium-quality VR over LTE. `LTE-VR` leverages cross-layer design and rich side channel information to reduce various latency sources in the signaling operations. Our prototype evaluation has confirmed its viability in 4G LTE. We discuss its applicability to the upcoming 5G.

CCS Concepts: • **Networks** → **Network protocol design**; **Network performance evaluation**;

ACM Reference Format:
Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, and Songwu Lu. 2018. Supporting Mobile VR in LTE Networks: How Close Are We?. *Proc. ACM Meas. Anal. Comput. Syst.* 2, 1, Article 8 (March 2018), 31 pages.
https://doi.org/10.1145/3179411

1 INTRODUCTION

In recent years, we have witnessed a boom in virtual reality (VR). 21 million wearable VR headsets are projected to be shipped in 2017, resulting in $4.9 billion revenue [66]; Google Cardboard alone has seen 160 million VR app downloads until February 2017 [21]. Among all the options, the mobile VR empowered by phones (e.g., Google Cardboard [29] and DayDream [30], Samsung Gear VR [62]) is most popular, contributing 98% of the sales [20]. Despite at early stage, it appeals to the general public with low cost (~$100) and excellent convenience (no wiring).

Mobile VR aims to offer users *ubiquitous* and *high-fidelity* experiences. If achieved, users can access VR "anytime, anywhere", regardless of whether they roam (e.g., in cars or on train) or remain static in indoor/outdoor settings. They also receive smooth, high-resolution panorama views (e.g., ≥60 frames per second and ≥1080p resolution) throughout the VR experience. It thus demands

Authors' addresses: Zhaowei Tan, University of California, Los Angeles, CA, USA, tan@cs.ucla.edu; Yuanjie Li, University of California, Los Angeles, CA, USA, yuanjie.li@cs.ucla.edu; Qianru Li, University of California, Los Angeles, CA, USA, qianruli@cs.ucla.edu; Zhehui Zhang, University of California, Los Angeles, CA, USA, zhehui@cs.ucla.edu; Zhehan Li, University of California, Los Angeles, CA, USA, zhehan@g.ucla.edu; Songwu Lu, University of California, Los Angeles, CA, USA, slu@cs.ucla.edu.

2476-1249/2018/3-ART8 $15.00
https://doi.org/10.1145/3179411

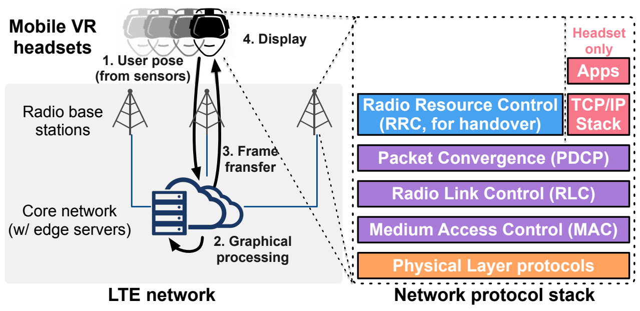


Fig. 1. Mobile VR over 4G (left); LTE protocol stack (right).

high bandwidth and stringent end-to-end latency in order to synchronize the graphical displays with the user motions.

A promising approach to enabling ubiquitous mobile VR is the edge-based scheme over 4G LTE networks [47, 56, 58]. As shown in Figure 1, the VR headset (smartphone) reports sensory user motions to edge servers wirelessly through the LTE network. The edge servers accept user input and deliver the requested graphics. They thus offload computation-intensive processing tasks from the battery-powered user devices. Ubiquitous access is provided by the LTE network, the only large-scale wireless infrastructure offering universal coverage and seamless mobility[1].

In this work, we examine several common perceptions, and study medium-quality mobile VR (say, 60 frames per second and 1080p resolution) over operational LTE networks. We show that, contrary to common understandings, bandwidth tends to be *not* the main bottleneck for medium-quality VR. Instead, network latency poses the biggest obstacle for the mobile VR support. Interestingly, a bulk portion of network latency does not stem from wireless data transfer, but comes from the *signaling operations* used by LTE to facilitate the wireless data delivery. These operations exhibit two categories of latency deficiency: (1) *Inter-protocol incoordination*, in which problematic interplays between protocols unnecessarily incur delays; (2) *Single-protocol overhead*, in which each protocol's signaling actions unavoidably incur delays. Along both dimensions, our analysis, together with 8-month empirical studies over 4 US mobile carriers, looks into five common beliefs on LTE network latency under both static and mobile scenarios (§3) and shows that they are wrong. In fact, they pose as roadblocks to enable mobile VR. Our three findings are centered on three existing mechanisms for data-plane signaling, which are all well known in the literature [5, 7, 42]. However, their deficiencies have not been studied from the latency perspective, particularly for delay-sensitive mobile VR applications (§5.1, §6.1, §6.2). We further describe a new finding that incurs long latency but has not been reported in the literature (§5.2). Moreover, we quantify the impact of each finding under VR traffic.

We devise LTE-VR, a device-side solution to mobile VR without changing device hardware or LTE infrastructure. LTE-VR adapts the involved signaling operations to be latency friendly, while being 4G standard compliant. In a nutshell, LTE-VR both reactively mitigates the unnecessary latency sources among protocols and proactively masks unavoidable latency inside each protocol. It exploits two ideas. First, it applies *cross-layer design* to both ensure fast loss detection and recovery and minimize VR duplicates during handover. Second, LTE-VR leverages rich side-channel information, which is only available at the device, to reduce the VR-perceived latency.

We have prototyped LTE-VR in a testbed with USRP and OpenAirInterface [52]. Our evaluation shows that, LTE-VR reduces the frequency of graphical frames that miss the human's tolerance by 3.7× on average. It meets the user's delay tolerance with 95% probability. It also achieves latency

[1]In contrast, WiFi offers small coverage and limited mobility support.

reduction comparable to 10× wireless bandwidth expansion. Furthermore, LTE-VR incurs marginal signaling overhead (5% more messages) and extra resource consumption (0.1% more bandwidth and 2.3% more radio grants).

We further note that our findings would probably carry over to the upcoming 5G system, since its working groups [9] are discussing to reuse the data signaling designs from 4G (§11).

2 SUPPORTING MOBILE VR IN 4G LTE

We consider the following usage scenario. Alice is attending her friend's commencement on a university campus. She is wearing her VR headset for the live commencement event and the realtime in-situ campus tour, as she sits through the ceremony, or walks around for a quick campus tour. After the commencement, she plays VR games or accesses VR social networks (e.g., Facebook Spaces [25]) as she sits on the passenger's side. She thus gains ubiquitous VR experiences.

The high-fidelity, ubiquitous VR experience can be offered through the edge-based, mobile VR system (Figure 1). It has three components: the mobile VR headset worn by the user, the edge server, and the LTE network. The headset uses the smartphone as the self-contained sensor and display. It tracks the runtime user pose via built-in sensors and reports it to the edge server. The edge processes graphical tasks given the user motion, and streams frames back to the headset for display. This involves the communication between the headset and the edge via 4G LTE.

High-fidelity VR experience. To ensure medium quality playback, the VR graphics should be rendered at ≥60 frames per second (FPS) and with ≥1080p resolution. However, the mobile VR headset has limited graphical capability and battery. The computationally heavy processing tasks are thus offloaded to the edge servers. The edge servers then quickly process and stream the frames based on realtime user motions.

We define the end-to-end VR latency T as the elapsed time from the user pose change to the corresponding frame display. Studies have shown that, human's tolerance of T is approximately ≤50 ms [14, 16, 18]. In our context, it includes network transfer latency and client/network processing latency. In practice, server-side processing usually takes ∼5 ms/frame [17]. Client-side frame decoding and rendering can take 6-12 ms [63] and ∼10 ms [26, 27], respectively. So the overall network latency should not exceed ∼**25 ms**.

Ubiquitous VR experience via LTE. To gain "anywhere, anytime" VR experience and offload the computation to the edge, the mobile VR headset needs wide-area wireless network access. To date, the only large-scale wireless infrastructure that offers universal coverage is the 4G LTE network. In a nutshell, LTE consists of the radio access network and the core network. The radio access network consists of base stations covering geographical areas, and provides wireless access to the mobile devices. The core network connects the radio access network to the edge and wired Internet. To access the network, the VR headset (phone) connects to a base station in its coverage area (called a cell). As the headset moves, the LTE network retains its seamless data service by migrating it to the neighboring base station. This procedure is called *handover*.

To facilitate data transfer and mobility support, the LTE runs a number of data signaling operations between the VR headset (phone) and the base station (Figure 1). Specifically, the physical layer enables wireless communications, the link layer offers reliable, in-order data transfer over the radio link, and radio resource control (RRC) defines the handover function.

3 LATENCY OF MOBILE VR OVER 4G LTE

We study network latency for edge-based mobile VR over LTE networks, and focus on medium-quality mobile VR (60FPS and 1080p resolution). We seek to understand three issues:

(Q1) Does 4G LTE have the potential to enable such a mobile VR?

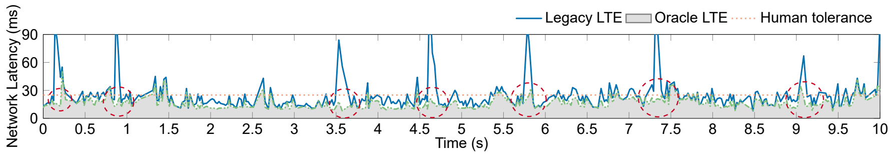


Fig. 2. Example of mobile VR's network latency under good LTE signals (-90dBm).

Category	Scenario	Misunderstanding	Problem	Section
Wireless bandwidth	Static/Mobile	M1	P1: Sufficient bandwidth for medium-quality VR	§4
Inter-protocol incoordination	Static	M2	P2: Delayed error recovery with two-tier retransmissions	§5.1
	Mobile	M3	P3: Head-of-line blocking by duplicates	§5.2
Single-protocol overhead	Static	M4	P4: Latency-unfriendly control channel designs	§6.1
	Mobile	M5	P5: Long disruptions in hard handover	§6.2

Table 1. Overview of latency components in LTE's signaling protocols.

(Q2) What are the roadblocks that impede mobile VR over LTE?
(Q3) What are the possible solutions to the discovered roadblocks?

Our work starts with the observation that the operational LTE network exhibits signs to meet the latency requirements by medium-quality VR. Figure 2 plots the trace of network latency for VR in a walking test with good radio coverage (~-90dBm). On the one hand, the legacy LTE indeed meets the human's latency tolerance (25 ms in §2) in many cases: 83% of VR frames arrive within the latency threshold. On the other hand, users still regularly perceive large VR frame delays. On average, one out of every 5.88 consecutive frames exceeds the human tolerance. For comparison, our solution reduces this frequency to one out of every 21.74 consecutive frames (in §9), approaching the Oracle LTE, which assumes global knowledge on LTE (further described in §9).

Five common misunderstandings on VR latency in LTE To better understand various latency causes and solutions, we examine five intuitively held beliefs for VR over operational LTEs:

M1: Wireless bandwidth is the bottleneck for the VR's network latency. Transmission time thus dominates the overall delay.
M2: When being transmitted wirelessly, the delivered VR data may be corrupted. 4G LTE can quickly recover the data.
M3: As the VR user moves, the device may reconnect to a new base station via *handover*. The device will immediately receive new data from the new base station thereafter.
M4: The user motion tracked by the VR sensors will be quickly sent out to the edge server for processing.
M5: When the device hands over to a new base station, 4G LTE will ensure unnoticeable latency.

Roadmap for the findings Surprisingly, our study shows that none of the above common beliefs holds. Wireless bandwidth is not the latency bottleneck for the medium-quality VR, thus invalidating M1. The wireless VR data transfer time only contributes to a small part of the overall network latency on average.

Instead, the LTE's *signaling operations* constitute a bulk portion of network latency. Such signaling functions intend to facilitate wireless data transfer (e.g., MAC and LINK protocols) and device mobility (handover). However, they incur long latency under various scenarios. Table 1 summarizes the identified issues. They arise regardless of signal strengths, time of experiments, phone models, VR applications, or competing traffic. They exhibit under both stationary and mobile scenarios, and can be divided into two classes:

1. Inter-Protocol Incoordination Signaling protocols are expected to work together for proper data delivery. However, their interplays can be problematic and slow. These latencies turn out to be unnecessary, and could be avoided *without* changing the LTE design.

• **Stationary scenario: Prolonged error recovery (§5.1).** During wireless transfer, LTE fails to quickly detect the corrupted VR data, and incurs long latency for recovery (M2). It is caused by the interplay between medium access control (MAC) and radio link control (RLC): The MAC's 1-bit stateless, unreliable feedback may not detect errors at the first place. Detection thus proceeds to higher-layer RLC and triggers slow recovery. The failure in error detection due to two-tier retransmissions is reported before. However, its latency effect has never been studied and quantified.
• **Mobile scenario: Head-of-line blocking by duplicates (§5.2).** Upon roaming, the headset is expected to immediately receive the latest VR data after handover (M3). Unfortunately, our study invalidates this premise. The headset receives data duplicates after handover, which preempt new data delivery. This is caused by the interplay between RLC and handover (RRC), where RLC is handover unfriendly. This is a new finding, and we will show why this head-of-line blocking happens and quantify its impact.

2. Single-Protocol Deficiency Extra latency also stems from a single protocol on signaling operations.
• **Stationary scenario: Latency-unfriendly control channel (§6.1).** A new uplink VR packet is expected to be delivered immediately (M4). However, the LTE control channel design may block the delivery and incur more queueing delay. We study how this default LTE uplink design adds unnecessary latency to VR.
• **Mobile scenario: Long disruptions due to hard handover (§6.2).** Handover is expected to be seamless upon roaming (M5). In practice, however, LTE incurs long disruption (≥20 ms by design). This is due to hard handover, which first disconnects from the old base station and then connects to the new one. This disruption is standardized, and we look into how this affects VR.

Methodology We first analyze the standards [4, 5, 10], and derive the latency equations for protocols. We then quantify them with experiments in operational LTE networks. We conduct an 8-month empirical study (12/2016-07/2017) over four U.S. carriers (Verizon, AT&T, T-Mobile, and Sprint). We use three popular mobile VR headsets: Samsung Gear VR (with Galaxy Edge 7), Google Cardboard and Daydream (both with Google Pixel and Huawei Nexus 6P). We use VRidge [60] (on DayDream/Cardboard) and StreamTheater [32] (on Gear VR) as representative VR apps with 1080p, 60FPS game streaming. They have 1M downloads in total on Google Store (until October 2017). To run experiments on more phone models, we also devise a VR traffic emulator based on real VR apps. The workload traces were obtained from VR headsets with both real VR traffic and the emulated traffic. While some experiments use the emulated traffic, our results show that their patterns are similar to the actual VR patterns. We have not observed biases so far. More details on the VR traffic collection are in Appendix A.1, where we also discuss how the signal strength affects our findings. For the edge server, we use a Dell XPS desktop with Nvidia GeForce GTX 745 GPU and Nvidia SHEILD [51] software. Our scheme emulates edge processing that is being deployed at edge servers (e.g., Nvidia partners with LTE vendors to deploy GRID edge services [31]).

We collect three types of traces to quantify network latency: (i) the LTE network signaling traces with MobileInsight [44]; (ii) the phone-perceived signal strength with Android's built-in CellSignalStrengthLte [36]; and (iii) the TCP/IP traces on clients and servers with tcpdump. We follow the LTE standards [4, 5, 10], break down the network latency into several parts, and identify each part. We have collected 3,165,000 LTE signaling messages in static, walking (~0.5m/s), and driving (~30mph) scenarios with varying radio qualities (-120~-80dBm). Our traces are collected on both weekdays and weekends, mainly during 7AM to 1AM. We include a detailed description on the dataset and show that the findings are not affected by phone model or time of experiment in Appendix A.2, and elaborate on how to calculate latency in Appendix A.3. We also validate some

VR App	Game	UL Pkt Interval (ms)	UL Pkt Size (KB)	UL Throughput (Mbps)	DL Throughput (Mbps)	DL Frame Size (KB)
VRidge	Airborne	14.4 ± 1.9	0.142	0.079	5.2	10.8
	Sleep Tight	14.5 ± 2.2	0.142	0.078	4.1	17.1
	Stacks 2.0	14.5 ± 2.0	0.142	0.078	9.2	19.2
Stream-Theater	Portal 2	17.7 ± 4.1	0.086	0.039	9.0	18.8
	CS GO	17.7 ± 4.2	0.086	0.039	9.1	19.0
	Civilization	18.3 ± 4.9	0.086	0.038	8.8	18.3

Table 2. Statistics of traffic patterns for 6 mobile VR apps.

findings using traces from MobileInsight's open database [48], which includes 67,285,000 cellular messages collected over two years.

4 IS BANDWIDTH THE BOTTLENECK?

We first discover that, simply expanding wireless bandwidth and improving radio signal strength may not suffice to meet the network latency requirement of VR (M1).

4.1 Network Traffic in Mobile VR

The network traffic from mobile VR is regulated by the VR headset and its control loop (§2), which are independent of network operators and radio signal strength. To characterize the traffic, we conduct experiments by running multiple mobile VR games (6 games with VRidge [60] and StreamTheater [32]). Figures 3 plots representative traces[2], and Table 2 shows the statistics.

We make three observations. First, the uplink VR traffic is small and periodic. The packets are small, since they usually only carry the user motion control event. Figure 3a shows that, the average device-side uplink packet arrival rate is only 38.8 Kbps. They are also periodic, because the built-in sensors at phones have constant sampling rate [22], regardless of the network technology. Second, the downlink traffic is modest for medium-quality mobile VR (60FPS, 1080p). Typical VR apps encode their downlink streaming data as video frames (e.g., H.264 in popular VR apps [32, 55, 60]), whose delivery is well supported by 4G LTE. Third, non-VR background traffic is negligible. When running the mobile VR app, other apps are suspended by the mobile OS, thus unlikely to send/receive significant network traffic. For example, our 1-hour experiment shows that, the background traffic accounts for 1.3% of total uplink traffic (2,051 out of 156,277 packets), and 0.04% of total downlink traffic (1,316 out of 3,038,965 packets).

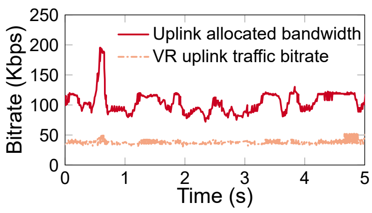
4.2 Enough Bandwidth for Medium-Quality VR

In practice, we find that the operational 4G base stations usually allocate more uplink radio resource than what medium-quality VR needs. Figure 3a shows one representative trace. The grants allocated by the base station are 103.8 Kbps on average, which is more than the average uplink packet arrival rate (38.8 Kbps). Similarly, the LTE downlink bandwidth is also sufficient for medium-quality VR. As shown in Figure 3b, the VR (60FPS, 1080p) requires 9.0 Mbps downlink rate on average. This is 1.9× smaller than the LTE downlink bandwidth (17.4 Mbps until November 2016 [53]).

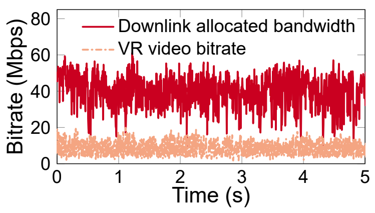
4.3 Is Bandwidth the Latency Bottleneck?

Wireless bandwidth is thus not the latency bottleneck for mobile VR. Table 3 shows the statistics of uplink/downlink data transmission time under good radio signal strength (-90dBm). For downlink, it takes 7.7 ms, 9.9 ms, 15.2 ms, 13.1 ms on average for AT&T, T-Mobile, Sprint and Verizon,

[2]This is a static case with radio signal strength around -95 dBm. We play VR game Portal 2 in StreamTheater, using Samsung S7 Edge for an Oculus Gear VR headset.



(a) Uplink



(b) Downlink

Fig. 3. Operational LTE networks offer sufficient bandwidth for medium-quality VR.

Network Operator	Uplink (ms)	Downlink (ms)
AT&T	0.1	7.7
T-Mobile	0.4	9.9
Verizon	0.1	13.1
Sprint	0.1	15.2
All	0.2	11.5

Table 3. Average data transmission time under -90dBm.

respectively, to transmit a video frame to the device. For uplink, all tested operators' uplink data transmission time contributes no more than 1 ms of network latency. Of course, for mobile VR with higher qualities (e.g. 4K 360 video), more wireless bandwidth is necessary to reduce the transmission time. The refined LTE technology (e.g., carrier aggregation [8]) can offer more bandwidth to meet this requirement. However, simply increasing wireless bandwidth still does not suffice to meet the latency requirement.

The above experiments are conducted under good radio signals (-90dBm). The results still hold if the radio signal strength varies (between -70dBm and -110dBm, see Appendix A.2). As the signal strength further degrades (≤-110dBm), transmission time may dominate the mobile VR's network latency. This situation could be significantly reduced with the emerging small-cell deployment.

5 INTER-PROTOCOL INCOORDINATION

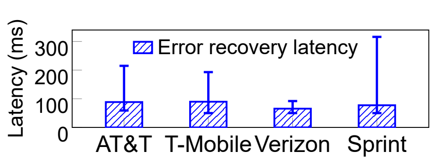
We now study the deficiency from improper protocol interplays.

5.1 Prolonged Error Recovery

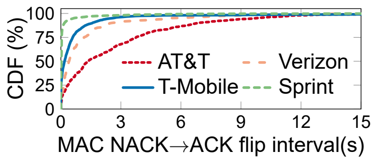
Different from M2, we discover that VR suffers from long error recovery delay. Figure 4a[3] shows the statistics from 4 operators. The average (maximum) recovery delays are 88.5 ms (215 ms), 89.7 ms (193 ms), 65.4 ms (316 ms) and 77.3 ms (92 ms) in AT&T, T-Mobile, Sprint and Verizon, respectively. Even with good radio quality (~-88 dBm), it still happens every 1.2s on average[4]. Such prolonged recovery is *not* caused by weak radio quality or network congestion. Instead, it is from the problematic interplays between Medium Access Control (MAC) and Radio Link Control (RLC) protocols.

Unreliable 1-bit error feedback at MAC. The LTE MAC detects data corruption via CRC, and issues a negative acknowledgment (M-NACK) for retransmissions. Upon no errors, an acknowledgment (M-ACK) is sent to ask for the new data. The acknowledgement signaling uses *1-bit* coding (0

[3]The error bar means the max and the min extra delay for each operator.
[4]It is quantified by counting the events that the base station incorrectly recognizes a retransmission as a new data (using new data indicator [10]), detailed in Appendix A.



(a) Extra re-transmission latency due to delayed error recovery.



(b) MAC NACK→ACK flip intervals.

Fig. 4. Downlink delayed error recovery.

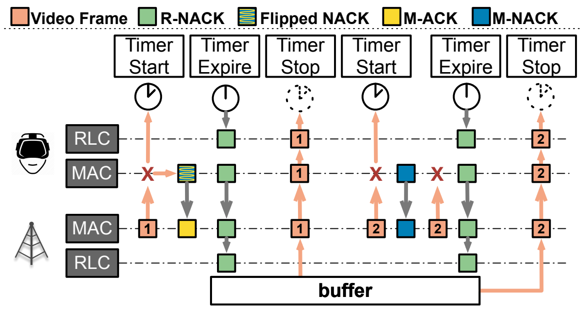


Fig. 5. LTE's two-tier retransmissions.

for M-ACK, 1 for M-NACK), and is delivered over separate physical-layer control channels[5]. These channels encode M-ACK/M-NACK and other signals in a compact form (e.g., binary physical-layer modulation or BPSK [10]) *without redundant bits*. The premise is that error feedback (M-NACK) could be quickly sent for fast recovery (M2 in §3).

However, this design could delay error detection. The 1-bit M-NACK can be flipped as M-ACK. Without redundant bits over the control channel, base station cannot detect this flip. The flipped bit will then be treated as M-ACK, and triggers the new packet delivery. MAC thus prolongs error recovery. The LTE design does not use redundant bits to save wireless bandwidth. Moreover, it assumes such flips are infrequent (10^{-3}–10^{-4} according to [42]). However, mobile VR presents an exception. The continuous downlink transfer amplifies the chances of M-ACK/M-NACK flips.

Delayed error recovery with two-tier retransmissions. Reliable, in-order data transfer is a common application demand. In LTE, MAC performs fast error recovery, and RLC ensures in-order delivery and further loss recovery. Both use retransmissions for error/loss recovery. Figure 5 exemplifies how they work. If a received VR packet is corrupted, its errors will be first detected and recovered by MAC. In case the errors cannot be recovered, RLC will take the responsibility. It treats this corrupted packet as a loss, and sends an RLC-layer NACK (R-NACK) for retransmissions. R-NACK is less frequent and more expensive than M-NACK.

Figure 5 illustrates how the error is propagated from MAC to RLC and triggers timeouts and long delay. For a corrupted VR downlink packet, a MAC-layer NACK is first sent. If this 1-bit M-NACK is flipped (1⟶0), the base station treats it as an M-ACK, and sends the next VR packet. Upon receiving the new packet, the device side MAC treats it as a new transmission, pushes it to the RLC layer, and thinks that the error is corrected. However, the corrupted packet is still missing at RLC. Note that LTE requires RLC to retain in-order delivery. An out-of-order event is thus triggered, and a retransmission timer $T_{reorder}$ is initiated (standardized in [8] and pre-configured). Upon timeout, RLC sends R-NACK and eventually recovers the corrupted packet. To ensure in-order delivery,

[5]In LTE, they are called Physical Uplink Control Channel (PUCCH) and Physical Downlink Control Channel (PDCCH) [11], respectively.

RSS (dBm)	[-60, -70)	[-70, -80)	[-80, -90)	[-90, -100)	[-100, -110)
Flip rate	0.0656%	0.0277%	0.0989%	0.5966%	0.9294%

Table 4. The radio signal strength's impact on flip rate.

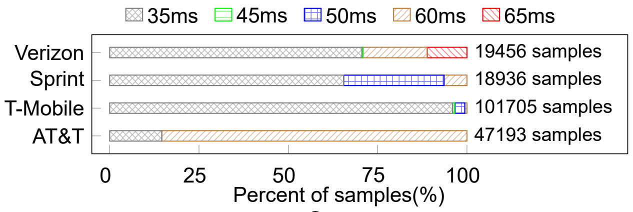


Fig. 6. $T_{reorder}$ **configuration statistics.**

RLC suspends all followup packets. Both the corrupted packet *and* the followup packets will be delayed by $T_{reorder}$.

Latency analysis. For the corrupted packets and the later ones, their downlink delay is prolonged by

$$T_{DL-retrans} = T_{reorder} + T_{RLC-UL} + T_{RLC-retrans} \tag{1}$$

$$T_{DL} = T_{DL-trans} + T_{DL-retrans} \tag{2}$$

where T_{RLC-UL} is the elapsed time to send R-NACK, and $T_{RLC-retrans}$ is the retransmission delay for the corrupted packet.

Validation. We quantify the frequency of MAC NACK→ACK flip in mobile VR. Figure 4b shows that, for all the operators, the NACK→ACK flip happens every 0.9 s on average. Table 4 shows the relationship between radio signal strength and flip rate. The flip probability tends to increase as the signal gets worse. However, such flip still happens even under good radio quality. For example, when the radio signal strength is -88 dBm, the flip happens every 1.2 s under our VR traffic.

We also find that, the prolonged error recovery delay is primarily determined by the timer $T_{reorder}$. Figure 6 summarizes the distributions of configured $T_{reorder}$ in reality. It shows that $T_{reorder}$ varies between 35ms to 65ms. Note that, 4G has standardized the configurable values of $T_{reorder}$ [4]. This explains why much longer delays are experienced on delayed error recovery.

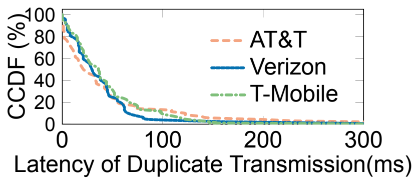
Uplink two-tier retransmission. Similar prolonged error recovery occurs for uplink transmissions, but it is rare and negligible. Our experiments show that, the RLC retransmission happens every 716.5 seconds. The reason is that, the VR's uplink traffic is small (§4.1). Fewer uplink packets will thus experience flips.

Design insight 1: *The MAC should not rely on 1-bit feedback only for error detection. Otherwise, it would unduly propagate errors to the higher-layer RLC, thus incurring large recovery latency.*

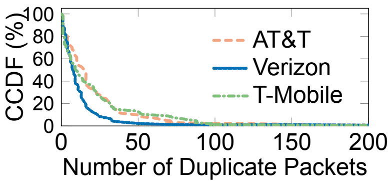
5.2 Head-of-Line (HOL) Blocking by Duplicates

M3 turns out to be also incorrect. After handover, the device receives duplicate packets that it has received before the handover. This causes HOL blocking. Before the client receives all duplicates, followup packets would be held for queueing. The duplicate delivery is common in reality. Depending on operators, 61.2% – 92.3% of handovers incur duplicate downlink packets (Figure 7[6]). Among the tested operators, this results in 30.0–44.7 ms HOL blocking latency on average, and 300 ms at maximum.

[6]Sprint suffers from similar issues. Its results are not included due to its implementation issue: It resets packet sequence number after handover, so duplicates are not counted.



(a) Head-of-line blocking latency



(b) Duplicate packets count

Operator	AT&T	Verizon	T-Mobile
Prob. dup. after HO	92.21%	92.31%	61.26%
Avg. # dup. packets	19.96	11.94	20.87
Avg. dup. delay (ms)	44.67	30.03	35.75

(c) Statistics of Duplicate Packets after Handover

Fig. 7. HOL blocking of duplicate packets after handover.

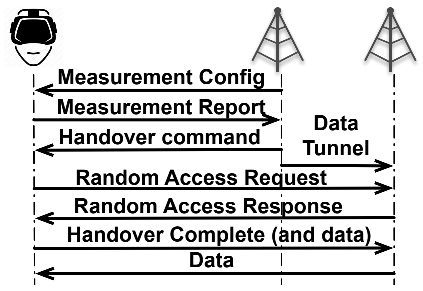


Fig. 8. Handover process.

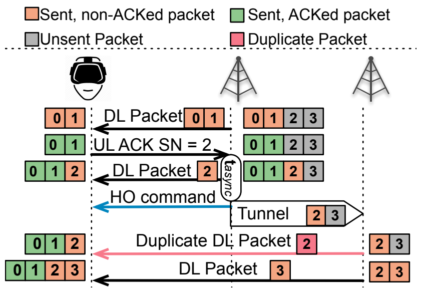


Fig. 9. Duplicate packets.

Operator	AT&T	Verizon	T-Mobile	Sprint
95% latency (ms)	3	2	2	5
Average latency (ms)	0.841	0.794	0.718	1.210
Maximum latency (ms)	11	5	24	18

Table 5. Latency of uplink head-of-blocking

This delay is caused by the problematic interplay between Radio Resource Control (RRC) and RLC. RRC handles the handover signaling and sets up forwarding tunnels (illustrated in Figure 8; details are in in Appendix B). However, RLC is handover unfriendly.

HOL blocking at downlink. RLC incurs duplicate packets and HOL blocking at downlink (exemplified in Figure 9). Before the handover, the device has successfully received some downlink packets from the old base station. However, it does not immediately respond with their acknowledgements (R-ACKs). The old base station thus tunnels these unacknowledged packets to the new base station. After the handover, the new base station retransmits the duplicates, although they have been received by the device. This incurs HOL blocking and prolongs the latency for new packets.

The client-side RLC has valid reasons not to acknowledge data immediately. The R-ACKs contribute to LTE signaling overhead. To mitigate it, RLC reduces the R-ACKs using timers (e.g., $T_{reorder}$ in §5.1) and polling (by the base station). Both mechanisms work well in the static case, without extra latency. For device mobility, however, they delay the acknowledgements and incur duplicates. As we will show in §7, with more client involvements, it is feasible to mitigate the HOL blocking with small signaling overhead.

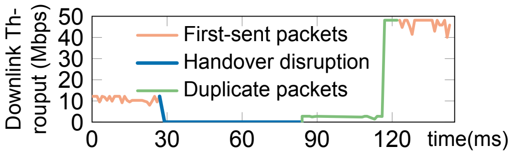


Fig. 10. Disruption & head-of-line blocking in handover.

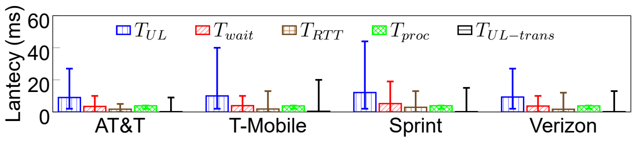


Fig. 11. Uplink latency breakdown in U.S. LTE operators.

Latency analysis. We next derive the HOL blocking delay in handover. We first count the duplicate packets as

$$n_{dup} = \frac{t_{async} * x_{old}}{L} \tag{3}$$

where t_{async} is the interval between the reception of last R-ACK (client-initiated) and the handover command (network-initiated). It characterizes the synchronization level. x_{old} is the average downlink goodput from the old base station during t_{async}, and L is the average VR packet size. When the device connects to the new base station, the new packets should wait until all duplicates are delivered. The HOL blocking latency is thus

$$t_{HOL} = \frac{n_{dup} \cdot l_{PDCP}}{x_{new}} = t_{async} \cdot \frac{x_{old}}{x_{new}} \tag{4}$$

$$t'_{DL} = t_{DL} + t_{HOL} \quad \text{(New packets only)} \tag{5}$$

i.e., it is proportional to the synchronization level (t_{async}) and the goodput from old (x_{old}) and new (x_{new}) base station.

HOL blocking at uplink. HOL blocking could also occur for uplink, but with negligible latency impact. Table 5 shows that, only ~1ms latency on average is incurred. For mobile VR, uplink goodput is much smaller (§4.1). Only few unacknowledged packets n_{dup} need retransmission. Given sufficient grants x_{new} after handover, following (4), latency by uplink duplicates is negligible.

Validation. We have validated and quantified HOL blocking in Figure 7 (explained above). We also find an extra issue that further prolongs t_{HOL}. We observe that, after handover, the downlink throughput cannot be immediately recovered. Figure 10 shows a trace. Similar issues are observed in all tested operators. After handover disruption (30–85 ms), the new base station first transmits data at an abnormally low rate. At t=20 ms, the downlink throughput recovers to 20× higher than the slow rate. According to (4), this delays the transmission of duplicates, thus prolonging t_{HOL}.

Design insight 2: *RLC should be handover friendly. It should speed up its data delivery feedback to the network before handover. According to (4), we shall decrease t_{async} to decrease the latency.*

6 SINGLE-PROTOCOL DEFICIENCY

We now elaborate on findings on extra latency incurred by an individual signaling protocol.

6.1 Latency-Unfriendly Control Channel

It turns out that M4 in §3 is also not true. When the VR headset samples the user pose change, the data cannot be immediately sent out to the edge. Table 6 shows that, 81.5% of VR uplink packets

Operator	AT&T	Verizon	T-Mobile	Sprint	Overall
Percentage	81.5%	79.1%	78.4%	87.2%	81.5%

Table 6. Percentage of packets that waited for SR

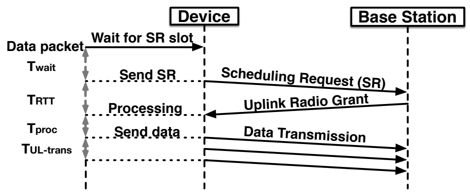


Fig. 12. Example of sending an uplink VR packet.

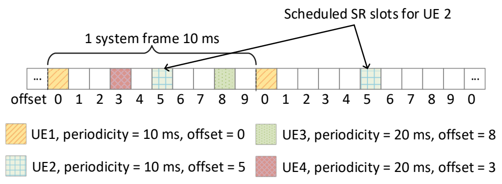

Scheduled SR slots for UE 2
1 system frame 10 ms
offset 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
UE1, periodicity = 10 ms, offset = 0
UE3, periodicity = 20 ms, offset = 8
UE2, periodicity = 10 ms, offset = 5
UE4, periodicity = 20 ms, offset = 3

Fig. 13. Periodical time slots for scheduling request (adopted from [10])

have to wait inside the phone, contributing to ~32.7% of VR latency in our experiments (Figure 11). This happens even when the uplink has sufficient bandwidth for data delivery.

Such queueing delay is caused by LTE's scheduling-based uplink control channel design. Figure 12 illustrates how an uplink VR packet is sent in LTE. The device first requests radio grants from the base station. It sends a scheduling request (SR) via the Physical Uplink Control Channel (PUCCH in Figure 13). Note that SR cannot be initiated anytime. The base station pre-allocates *periodic* physical-layer slots for each device to send SR, and notifies the device during the radio connectivity setup[7]. The device holds SR until its own slot is available. The uplink data is thus delayed.

The LTE scheduling and periodic SR prefer channel utilization over latency. To save radio bandwidth, the base station only allocates the grants based on device demands. The uplink control channel also shares the underlying physical resource with the data channel. If signaling messages (e.g., SR) use the physical resource, uplink data cannot be delivered. For high efficiency, the LTE pre-allocates periodical slots for SR, and leaves others for data.

At first glimpse, such utilization-latency tradeoff seems unavoidable. But for VR, *both T_{wait} and T_{RTT} could be masked if the client plays a more active role.* The key is that, the VR's uplink packet arrival is *periodic* (§4.1). The device could predict the next uplink packet's arrival, and proactively request the grants. The base station cannot do it, since it has no information on *when* the client sends the next uplink data. We will elaborate it in §7.1.

Latency analysis. The above analysis shows that, the total uplink latency T_{UL} for each VR uplink packet is

$$T_{UL} = T_{wait} + T_{RTT} + T_{proc} + T_{UL-trans} \tag{6}$$

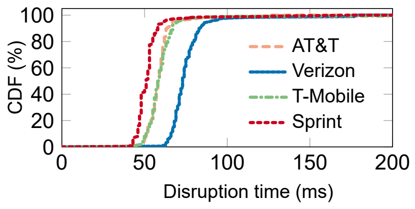
where T_{wait} is the elapsed time to wait for the SR slot (bounded by SR-ConfigIndex, a period pre-configured by the base station. The standard regulates its value between 1–80 ms [5]), T_{RTT} is the round-trip time between SR and the grant notification, T_{proc} is the device's local processing delay (upper bounded by 4 ms [5]), and $T_{UL-trans}$ is the transmission delay. Ideally, the uplink delay T_{UL} should only involve the processing and transmission delay ($T_{proc} + T_{UL-trans}$). However, according to (6), extra delays are caused by waiting for SR T_{wait} and the round trip time T_{RTT} for grant allocation.

Validation. We first quantify the extra delay $T_{wait} + T_{RTT}$ perceived by uplink VR packets. We collect 671,063 VR uplink packets from four U.S. operators under varying radio quality, and perform the breakdown analysis of uplink latency T_{UL}. Figure 11 shows that, the average $T_{wait} + T_{RTT}$ are 6.3 ms, 7.0 ms, 9.0 ms and 6.6 ms for AT&T, T-Mobile, Sprint, Verizon, respectively. In Table 6, we

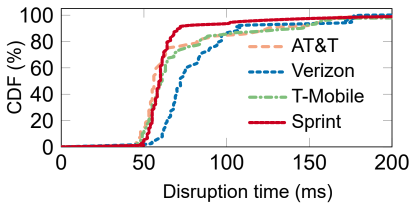
[7]It is encoded as SR-ConfigIndex in RRCConnectionReconfiguration message [8].

Length	5 ms	10 ms	20 ms	40 ms	80 ms	Sample #
AT&T	0	44229 (90.3%)	4737 (9.8%)	8 (0.0%)	0	48975
T-Mobile	3 (0.0%)	103207 (97.2%)	2638 (2.5%)	346 (0.3%)	3 (0.0%)	106198
Sprint	227 (1.1%)	10138 (50.5%)	5700 (28.4%)	3875 (19.3%)	152 (0.7%)	20093
Verizon	0	18387 (92.3%)	79 (0.4%)	1452 (7.3%)	0	19919

Table 7. SR-periodicity configuration samples.



(a) Disruption for uplink data.



(b) Disruption for downlink data.

	Verizon	AT&T	T-Mobile	Sprint
Uplink (ms)	75.74	59.43	60.62	52.59
Downlink (ms)	83.75	72.30	76.92	67.48

(c) Average handover disruption time.

Fig. 14. Handover disruption time.

show that 81.5%, 78.4%, 87.2%, 79.1% of their observed uplink packets have non-zero $T_{wait} + T_{RTT}$. Note that the waiting delay T_{wait} is bounded by SR-ConfigIndex, and Table 7 presents statistics of SR-ConfigIndex. This period varies between 5ms to 80ms among base stations and operators. The remaining 18.5% packets do not experience $T_{wait} + T_{RTT}$ since grants are already available for them.

Design insight 3: *The LTE's uplink control channel for resource allocation should be latency friendly. The latency-utilization tradeoff could be bypassed using the VR uplink traffic's regular pattern.*

6.2 Long Disruptions with Hard Handover

Another latency deficiency stems from handover, and invalidates M5 in §3. Figure 10 illustrates a LTE handover trace. It incurs 54 ms disruption (excluding the HOL blocking latency in §5.2), during which the VR's delay requirement cannot be met. Figure 14 shows that, the average disruptions are 83.7 ms, 72.3 ms, 76.9 ms and 67.4 ms in AT&T, T-Mobile, Sprint and Verizon, respectively.

The disruption in handover is rooted in LTE design. As shown in Figure 8, the LTE handover follows the *"break-before-make"* paradigm. The device first disconnects from the old base station, and then connects to the new base station. In between, the device cannot gain network service from either base station.

At first glance, the LTE design could prevent the disruption. A possible solution is *soft handover*, which follows the "make before break" strategy in 3G [3] and the device maintains concurrent accesses to both base stations, thus retaining always-available service. Unfortunately, the LTE's radio technology prohibits it. 4G LTE uses the Orthogonal Frequency-Division Multiplexing (OFDM) technology. Compared with 3G (using CDMA), it is hard (if not impossible) for OFDM to keep simultaneous connectivities to both base stations. 4G thus decides to not support soft handover.

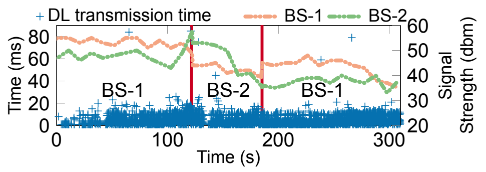


Fig. 15. A case of unnecessary handover.

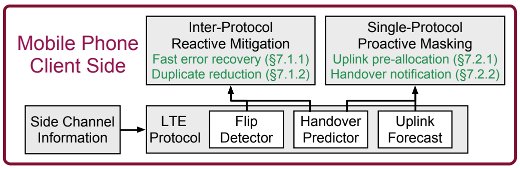

Mobile Phone Client Side
Inter-Protocol Reactive Mitigation
Fast error recovery (§7.1.1)
Duplicate reduction (§7.1.2)
Single-Protocol Proactive Masking
Uplink pre-allocation (§7.2.1)
Handover notification (§7.2.2)
Side Channel Information
LTE Protocol
Flip Detector
Handover Predictor
Uplink Forecast

Fig. 16. LTE-VR design overview.

Latency analysis. Following the standards [7], we derive the bound of the disruption time in LTE handover as follows:

$$T_{handover} \leq T_{device-proc} + T_{random-access} \tag{7}$$

where $T_{device-proc}$ is the time spent by the device to prepare for connecting to the new base station, and $T_{random-access}$ is the random access round-trip. $T_{device-proc}$ further has two parts:

$$T_{device-proc} = T_{search} + 20\ ms \tag{8}$$

where T_{search} is the scanning of the new base station. To guarantee sufficient time for local processing, [7] allows for 20ms safeguard interval. While reasonable for most apps, we discover that such choice is not VR-friendly: Given the VR's latency demand (≤25ms), such long disruption will incur delays.

Validation. We first validate and quantify the disruptions in LTE handover. Figures 10 and 14 show the existence and impact of disruption in LTE handover. We also find that, some handovers are unnecessary. Figure 15 shows one trace from our uni-direction walking tests. Two handovers occur between base stations 1 and 2. Before and after each handover, the downlink latency remains comparable. This implies that, such handovers do not help but cause unnecessary disruptions. These handovers are not triggered by the user movements. Instead, the sudden surge of base station 2 makes the device temporarily switch to it and incurs 2 handovers, even though base station 1 has good signal strength. In reality, our experiment shows that 17.5%, 15.0%, 31.0%, 24.6% of the handovers for Verizon, AT&T, T-Mobile and Sprint respectively, can be attributed within this category. They could be avoided to prevent disruptions.

Design insight 4: *Since LTE handover unavoidably incurs network disruptions, it is vital to steering clear of unnecessary handovers if possible, and also being prepared to embrace it before it occurs.*

7 LTE-VR: LTE BOOSTER FOR MOBILE VR

We devise LTE-VR, a client-side LTE solution to edge-based mobile VR. LTE-VR seeks to achieve two goals: (1) to mitigate network latency for mobile VR; (2) to be readily available to VR headsets without any infrastructure changes and be standard compliant.

Figure 16 illustrates LTE-VR's design. To meet the above goals, it adapts the client-side signaling protocols, without requiring network support or modified standards. It leverages the insights in §4–6, and pursues two dimensions: (1) reactively mitigate the unnecessary latency from inter-protocol incoordination, and (2) proactively mask the unavoidable latency from single-protocol actions.

The key premise is that, *only the client has the essential information for latency mitigation/masking*. LTE-VR leverages rich *side-channel* information, which is only available at the device. Such side-channel information includes: (i) standardized differential processing [10] of 1-bit M-ACK/M-NACK; (ii) handover measurement reports; and (iii) sensory data. Note that all three types of information can only be available at the phone. Moreover, it uses cross-layer design to ensure fast error recovery and minimize duplicates in handover. LTE-VR retains signaling and resource overhead similar to LTE, thus ensuring the same 4G scalability to VR users.

7.1 Reactive Inter-Protocol Latency Mitigation

LTE-VR first mitigates the latencies due to the delayed error recovery (§5.1) and the HOL blocking by duplicates (§5.2).

7.1.1 Static: Accelerate Error Recovery. LTE-VR first reduces downlink retransmission latency by accelerating LTE's link-layer error recovery. It speeds up the MAC detection of M-NACK→M-ACK flip, and devises low-cost feedback for fast retransmission by following Insight 1.

Fast M-NACK/M-ACK flip detection. The phone-side MAC can perform fast detection, whereas the base station cannot. Without redundant bits, the base station cannot check if a 1-bit feedback is flipped (§5.1). Adding redundant bits on the control channel is not compliant to LTE. Instead, with local M-ACK/M-NACK states, the phone can detect the flip accurately.

Specifically, LTE-VR runs timer-based flip detection. Note that, the base station takes *shorter* time to process M-NACK than M-ACK *by LTE design* [5]. It processes M-NACK with higher priority than M-ACK to immediately retransmit the corrupted data. LTE-VR leverages this to detect the flip. It maintains and updates a phone-side MAC timer T_{M-NACK} for the round-trip of M-NACK, based on previous M-NACK-triggered retransmissions using exponential moving average: $T_{M-NACK} \leftarrow \frac{7}{8}T_{M-NACK} + \frac{1}{8}T_{retrans}$. When an M-NACK is sent, the timer starts. If no response before the timer expires, LTE-VR recognizes that the base station may receive a flipped M-NACK as M-ACK. Indeed, this approach may have false-positives due to inaccurate timer estimation. But the cost is marginal: At most one retransmission would be incurred with ~1 ms extra latency.

Fast feedbacks for retransmission. Given the M-NACK→M-ACK flip notification, LTE-VR requests the network to retransmit the corrupted packet *immediately* to accelerate the recovery. Ideally it should be done at the phone-side MAC. Unfortunately, this does not work. Upon receiving the flipped 1-bit M-ACK, the network-side MAC would infer that the corrupted packet has been correctly delivered, and drop it from the MAC buffer. Instead, LTE-VR requests retransmissions at RLC, where R-NACK indicates the sequence number of the corrupted packet.

7.1.2 Mobility: Mitigate HOL Blocking. LTE-VR makes RLC handover friendly using Insight 2. By accelerating R-ACKs/R-NACKs, the phone quickly notifies the network of VR data reception *before* the handover event, thus reducing duplicates and HOL blocking. To keep the signaling overhead low, LTE-VR selectively enables the fast feedback.

Lightweight, handover-friendly reaction. Given that HOL blocking only happens in handover, LTE-VR activates fast feedback only when handover is about to occur. This is realized by predicting the handover event (detailed in §7.2.2) and switching to the fast-feedback mode. Note that the handover prediction is not perfect: If a handover were predicted but does not occur, LTE-VR retains the same latency as 4G LTE. If a handover is missed, extra R-ACK/NACKs may be triggered; LTE-VR deactivates the fast feedbacks if handover does not happen as predicted.

Fast feedbacks for duplicates mitigation. LTE-VR replies R-ACKs/R-NACKs *immediately* on receiving the link-layer packets, without waiting for polling requests or timeouts. This prevents the old base station to tunnel the received packets to the new one. This mode is only enabled before the handover, so its signaling overhead is thus marginal (in proportion to the duplicates).

7.2 Proactive Single-Protocol Latency Masking

LTE-VR masks the network latency incurred by each protocol for VR apps. It tackles the waiting delay for uplink radio resource (§6.1), and the disruption during the handover (§6.2).

7.2.1 Static: Proactive Radio Resource Allocation. LTE-VR first adapts the LTE's uplink resource control to be latency friendly (Insight 3 in §6.1). To mask the latency, LTE-VR proactively requests

ALGORITHM 1: Adaptive scheduling request (SR) activation.

```
Input: Current time slot, SR-ConfigIndex, estimated $T_{RTT}$ and $T_{interval}$
Output: Whether an SR should be sent in this time slot
1 Calculate next packet arrival, based on $T_{interval}$ and last packet arrival
2 if Current time slot can send SR then return (Device side buffer > 0 ∨ Predicted next uplink packet arrives in
    $T_{RTT}$);
3 return false otherwise
```

uplink radio resource *before* uplink VR packet arrival to the LTE stack. The VR packet can then be sent without waiting.

The challenge is to avoid wasting the pre-allocated radio resource. As shown in §6.1, this seems to be a fundamental conflict between utilization and latency. Fortunately, it can be resolved by exploiting the VR traffic regularity (§4.1). LTE-VR predicts the VR packet arrival, and adapts the scheduling request to minimize the potential waste.

Predicting the uplink packet arrival. LTE-VR predicts the next packet arrival by (a) estimating the inter-packet arrival time $T_{interval}$, or (b) using the periodicity configuration from the VR sensor (if available). It observes the recent uplink packet arrival time, and updates $T_{interval}$ estimation as follows: $T_{interval} \leftarrow \frac{7}{8}T_{interval} + \frac{1}{8}T$, where T is the latest inter-packet arrival time. It is accurate in mobile VR (validated in §9), because of the periodic VR traffic and negligible background traffic. This approach incurs marginal overhead since (1) only variable $T_{interval}$ is maintained; (2) to monitor the packet arrival, it reuses the LTE's mechanism without extra overhead. If the VR pose sensor reports its sampling rate, more accurate estimation with negligible overhead is possible.

Adaptive LTE scheduling request. Algorithm 1 illustrates how LTE-VR adapts the scheduling request for uplink radio resource. When a time slot is ready, LTE-VR checks whether the next uplink packet can be sent if SR was sent in this slot. If true, an SR is sent. This avoids the potential waste from the pre-allocated resource.

On background and non-periodic traffic. LTE-VR may also reduce latency by pre-requesting the resource. Note that the background traffic incurs extra overhead. With both VR and background traffic, $T_{interval}$ may be under-estimated, thus causing more requests and over-allocation. This is infrequent: The background traffic is negligible when running VR on the phone (§4.1). For non-periodic traffic, LTE-VR can detect such non-periodicity by tracking the variance of $T_{interval}$. It can thus disable the proactive allocation and prevent the resource waste.

7.2.2 *Mobility: Mask Handover's Disruption via Prediction.*

LTE-VR seeks to mask handover disruption for VR (Insight 4). It devises accurate handover prediction ahead of its occurrence (~100 ms ahead), thus allowing proactive adaptation *before* the handover.

Handover prediction. We note that, the LTE handover decision logic is *interactive, stateful and stable*. Figure 8 illustrates how the base station configures the device with the standardized criteria [8] of measuring the current and nearby base stations' radio signal strength (Appendix B). The device measures and reports to it if any criteria is satisfied. The base station then determines whether to run the handover based on a pre-determined algorithm. Such mechanism has been widely used [1, 2]. While different base stations may have varying configurations for the thresholds of triggering these criteria, the decision logic largely remains invariant. By mapping the measurements (reports to network) to the handovers (response from network), the device can infer the base station's decision algorithm and predict the handovers.

LTE-VR devises a lightweight predictor (Algorithm 2) based on our experimental data and popular base station's internal decision logic [1, 2]. It classifies the handover into two categories: *Intra-frequency* handover using the same frequency band, and *inter-frequency* handover using different

ALGORITHM 2: LTE handover prediction.

```
Input: Serving cell $s$. Measurement report sequence $(m_1, m_2, ..., m_k)$ where $m_i$ is the signal strength of cell $i$.
Output: Boolean indicator of handover.
1 if s.freq=$m_1$.freq and $m_1$.event=A3, where s.freq is the frequency band of s and event is defined in Table 14 then
    return true;
2 else if $s$.freq$\neq m_1$.freq and $s$.event=A2 and $m_1$.event=A5 then return true;
3 return false otherwise;
```

frequencies. In reality, they are triggered by different criteria. The intra-frequency handover is triggered when a nearby base station's signal strength is better than the current one (event A3 defined in Appendix B). Otherwise, Inter-frequency handover is triggered when the current base station's signal strength is weak (event A2 in Appendix B), and a nearby base station's signal strength is satisfactory (event A5 in Appendix B). LTE-VR monitors these events to predict the handover, which offers high accuracy in reality (§9.2). This result is insensitive to threshold configurations across base stations. Our algorithm, as well as the inferred decision logic, is based on measurement events.

Prediction-enabled latency masking. The handover prediction helps mask latency at both network and app levels. It mitigates the unnecessary handovers at the network level. It further notifies the VR app for early adaption (e.g. pre-rendering [19, 43] and FPS adaption). This notification can be sent via OS APIs or the diagnostic mode (e.g., with MobileInsight [48]).

Prediction accuracy analysis. The high accuracy of our prediction is rooted in its (1) approximation to base station's handover decisions; and (2) robustness to noisy wireless channels. We next analyze each, and discuss how false positives/negatives are tackled.

∘ *Approximation to network-side decision.* Our prediction follows the same mechanism and triggering conditions as the algorithms in most commercial base stations [1, 2]. Note that, the base station may indeed leverage other criteria for handover decisions (e.g., load balancing). Despite this, our evaluation in §9.2 shows that LTE-VR achieves $\geq$ 90% accuracy in all tested operators.

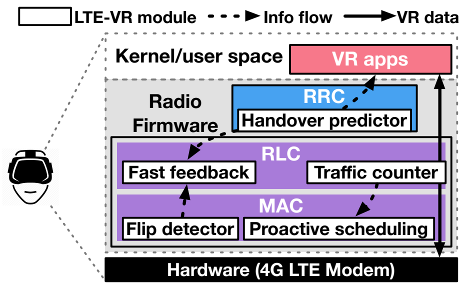
∘ *Robustness to dynamic/noisy wireless channels.* Our prediction is robust to channel dynamics and noises. Algorithm 2 is based on measurement-triggered *events*, rather than direct radio measurements. To handle transient dynamics, the standards [8] have defined thresholds for event triggering (in Appendix B). False measurements due to radio fluctuations/noises are thus mitigated, resulting in a robust prediction.

∘ *Tackling false positives/negatives.* If handover is mis-predicted, LTE-VR ensures that VR experiences latency no larger than 4G LTE. There are two scenarios. First, LTE-VR may predict a handover that would not happen. Both network and app-level adaptations do not incur extra latency or overhead. Fast feedback for retransmissions (§7.1.1) may be falsely triggered. The R-ACK/R-NACK overhead is bounded, since LTE-VR deactivates it once handover does not happen as expected (∼100ms based on evaluations in §9.2). Second, LTE-VR may miss a handover. This rolls back to the standard LTE.

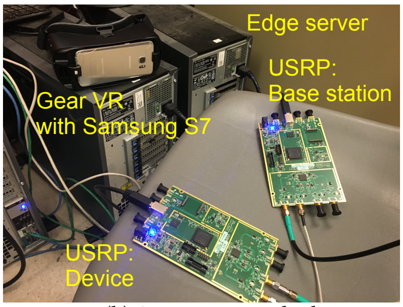
8 IMPLEMENTATION

Figure 17a shows our implementation as a client-side radio firmware patch (typically with OS upgrade[8]), and is generally applicable to various headsets/phone models. The low-level implementation may vary between OSes (Android/iOS) and chipsets (Qualcomm/MediaTek).

[8]For example, the Android factory images [28] include the firmware radio.img (customized by phone vendors or network operators).



(a) System overview.



(b) Prototype test-bed.

Fig. 17. LTE-VR prototype.

8.1 Reactive Latency Mitigation

Error Correction Accelerator (§7.1.1). It is prototyped in two steps. First, the fast M-ACK/M-NACK flip detector is implemented as a MAC extension. A timer T_{M-NACK} is kept and updated based on successful retransmissions, and used to estimate the flip by following §7.1.1. Second, LTE-VR realizes the fast retransmissions at RLC. With the MAC notification (via an event handler), we extend RLC's logic to initiate R-NACK without waiting.

HOL blocking mitigator (§7.1.2). We realize it at RLC and RRC. We first implement the handover predictor at RRC, and use it to notify RLC (as an event handler) when a handover is predicted. We then extend RLC with fast retransmissions. Given the handover prediction, it immediately replies R-ACK/R-NACK.

8.2 Proactive Latency Masking

Proactive radio resource allocation (§7.2.1). This is realized at MAC. To realize the uplink data arrival predictor, we maintain a variable $T_{interval}$, and update it upon new data. This needs access to the MAC buffer, which is readily available in LTE since it has an event handler that monitors incoming packets. For higher accuracy, we also allow the hardware pose sensors to relay their sampling intervals to MAC. For proactive allocation, we extend MAC logic of sending SR. When MAC decides whether to activate an SR, it decides whether this SR should be triggered based on Algorithm 1.

Handover predictor, mitigator and notification (§7.2.2). To predict handover events, we monitor the RRC's measurement reports from their event handlers, and run Algorithm 2. Then handover predictions shall be sent to the network layer and VR apps. We support notification to RRC itself (for unnecessary handover mitigation) at the network level. This is achieved by extending the RRC event handlers of measurement reports and blocking the report delivery if an unnecessary handover is predicted. For notification to apps, we can either add Android APIs, or leverage the diagnostic mode [48] to relay the information to apps.

9 EVALUATION

We assess LTE-VR's overall latency reduction for VR apps (§9.1), examine its components (§9.2), and quantify its overhead (§9.3).

Since the phone-side radio firmware source is not open to us (but available to phone vendors [59]), we approximate the implementation of §8 as an extension of the open-source LTE stack OpenAirInterface [50, 52]. A software-defined radio (USRP B210) (Figure 17b) runs as the VR phone, while others as the base stations.

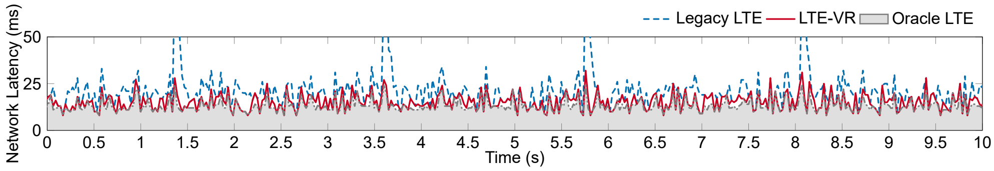
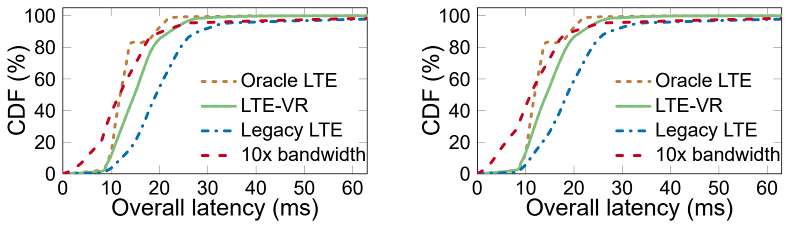


Fig. 18. Example of mobile VR's LTE latency with LTE-VR and LTE Oracle.

Application	Average Reduction	Maximum Reduction	95% Reduction	% Under 25 ms	Average # of frames between over 25 ms
VRidge	5.7 (26.5%)	70.0 (75.3%)	10.0 (46.1%)	82.9% → 95.4%	5.88 → 21.74
StreamTheater	5.4 (25.7%)	70.0 (77.8%)	10.0 (47.8%)	85.1% → 96.1%	6.71 → 25.64

Table 8. Overall latency reduction from legacy LTE using LTE-VR.



(a) VRidge (b) StreamTheater

Fig. 19. CDF of overall network latency.

9.1 Overall Performance

We assess how LTE-VR reduces network latency over two representative VR apps: VRidge and StreamTheater. We collect their network traffic in a driving test (20,000 1080p frames), and replay them in our testbed. Radio signal strength varies within [-95dBm, -105dBm]. We evaluate the network latency defined in §2, and set human's tolerance of the overall network latency as 25ms. We compare LTE-VR with three alternatives (detailed in Appendix C): Legacy LTE, Oracle LTE, and LTE with bandwidth expansion. Table 8 and Figures 18–19 summarize the results.

Overall latency reduction. For both VR apps, LTE-VR reduces their overall network latencies. As shown in Table 8, compared with legacy LTE, LTE-VR reduces overall latency by 5.7 ms (26.5%, from 21.3 ms to 15.6 ms) on average, and 70 ms (out of 93, 75.3%) at maximum. In legacy LTE, 1 out of every 5.88 frames (or every ~98 ms equivalently) exceeds latency tolerance. LTE-VR lowers this frequency by 3.7× (every 21.74 frames, ~360 ms). Moreover, 95% of the graphical frames in LTE-VR arrive within the user's 25ms tolerance (83% in legacy LTE). LTE-VR (95% of frames have satisfactory latency) approximates the oracle LTE (98% of frames have satisfactory latency), i.e., the lowest achievable latency over LTE.

Comparison with expanding wireless bandwidth. Figure 19 shows that, LTE-VR reduces similar latency to LTE with 10× bandwidth expansion (7.5 ms). Note that, upon the handover and delayed recovery events, expanding wireless bandwidth would not help. In this case, LTE-VR saves 36.5 ms more than bandwidth expansion.

9.2 Effectiveness of Key Components

We next evaluate the effectiveness of LTE-VR's key components.

Error Recovery Accelerator (§7.1.1). We first examine how fast LTE-VR accelerates error recovery. In this test, we inject random VR data corruptions and M-NACK→M-ACK flips based

$T_{reorder}$ (Original Recovery Time)	Recovery Time in LTE-VR	Extra R-NACK	Average Saving
60.0 ms	10.7 ms	4.3%	49.3 ms
35.0 ms	12.4 ms	8.2%	22.6 ms

Table 9. Benefits and overhead in error recovery accelerator.

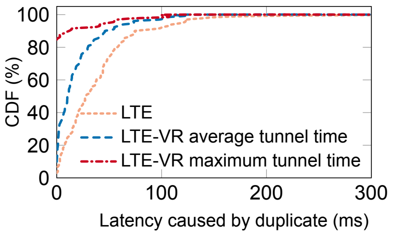


Fig. 20. LTE-VR's savings for head-of-line blocking.

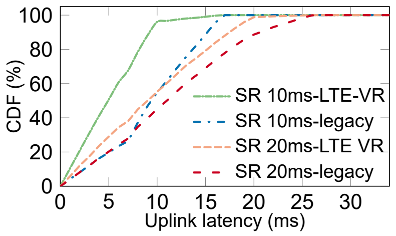


Fig. 21. LTE-VR's uplink latency reduction.

on user traces (every 1.2s on average, see §5.1), and gauge the recovery latency in legacy LTE and LTE-VR. We run tests with varying values of the RLC timer [8] (~5,000 flips), and repeat 10,000 runs for each.

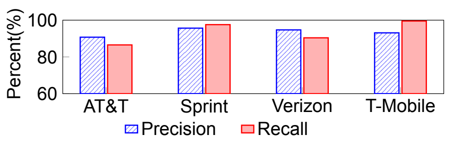
Table 9 shows the results. LTE-VR reduces the average time of recovery from 60ms (35ms, depending on $T_{reorder}$) to 10.7ms (12.4ms), with 49.3ms (82%) and 22.6ms (65%) reductions. It reduces maximum 52.0ms recovery delay. Moreover, LTE-VR saves more under larger $T_{reorder}$, which causes longer waiting in legacy LTE following (1) but does not affect LTE-VR. We have also tried various averaging methods for timer updates, but do not see major differences. The exponential average (§7.1.1) also approximates the latency bound.

HOL blocking mitigator (§7.1.2). We next assess how well LTE-VR mitigates the HOL blocking delay in handover. Since handover is not fully realized in OpenAirInterface, we take the emulation approach: We replay all handover events (50 hours in total) from our LTE traces, and assess the duplicates and latencies. To compute the reduction of duplicate retransmissions, we count all PDCP packets retransmitted after handover, and assess how many can be mitigated with LTE-VR. According to (3), this requires knowing when the old base station creates the tunnel to forward unacknowledged packets. Unfortunately, such information is not directly accessible to the device. Following the standards [6], we estimate its upper bound as the interval between the last acknowledgement before handover and the time of handover. We assess the latency reduction using this bound and its average.

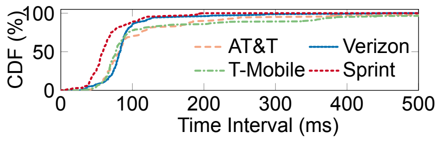
Figure 20 shows the results. On average, LTE-VR saves 52.0% duplicates and thus 20.80ms HOL latency if tunnel is established for the average case, and 88.3% duplicates and 34.8ms if the tunnel is established in the latest possible time.

Proactive radio resource allocation (§7.2.1). We quantify its reduction of waiting delays and its overhead for uplink VR packets. The setup is identical to §9.1, but we repeat it with different SR-ConfigIndex settings. We use more than 50,000 uplink samples. For each uplink packet, we compare its waiting delay in LTE-VR and legacy LTE. Figure 21 shows that, LTE-VR saves 4.4 ms (44.4%) per-packet uplink latency (9.9 ms to 5.5 ms) on average. The longer SR-ConfigIndex is configured, the more latency LTE-VR saves.

Handover predictor, mitigator and notification (§7.2.2). We evaluate the handover predictor. We apply Algorithm 2 to all measurement reports from our LTE traces, and evaluate it with three metrics: (1) *Precision*: The percentile of predicted handovers that have actually occurred; (2) *Recall*:



(a) Prediction accuracy



(b) Earliness

Fig. 22. Effectiveness of LTE-VR's handover prediction.

The percentile of occurred handovers that are successfully predicted; (3) *Earliness*: The interval between handover prediction and the handover occurrence.

Figure 22a shows that, among all operators, LTE-VR predicts 797 handovers in total, and achieves at least 90.7% precision and 85.6% recall. Figure 22b shows that, on average, it predicts the handover 100.9ms ahead of its occurrence, leaving sufficient time for latency masking. We have observed two factors that affect the prediction accuracy: (a) *Extremely weak wireless channel*: The base station may fail to receive measurement reports. The predicted handover may thus not occur; (b) *Difference of event handling:* Some AT&T base stations use inter-frequency handover once receiving A5 only (no A2). They incur more handovers that are not predicted by LTE-VR.

We next showcase how handover prediction helps mask VR latency. §6.2 has shown that 15.0%–31.0% unnecessary handovers can be mitigated among network operators, thus avoiding disruptions. At the app level, we assess how VR apps mask the latency by pre-rendering frames. We use the handover predictor on Google Pixel running StreamTheater, and count the video frames that can be delivered in the prediction interval (100.9 ms above). On average, our solution allows 7 frames to be pre-transmitted prior to handover. This can enable the phone to mask VR latency.

9.3 Low System Overhead

Signaling overhead. LTE-VR may incur extra signaling messages in two scenarios. First, accelerating the error recovery needs more R-NACK signaling feedbacks. This overhead is marginal. Table 9 shows that, it incurs 4%-8% more feedbacks than the legacy LTE. Second, to make RLC be mobility friendly, LTE-VR may trigger more R-ACK/R-NACK overhead. This is also marginal. On average, it incurs 5.0% more R-ACKs/R-NACKs than legacy LTE. Even if the handover is mis-predicted, its overhead is still bounded by 5.0%. Such low overhead facilitates similar scalability to VR devices as 4G LTE.

Radio resource overhead. LTE-VR may request more radio grants with proactive scheduling for uplink. Our test shows that, on average, only 2.4 Kbps extra grants would be allocated and wasted, accounting for 2.3% of the total grants (103.8 Kbps for uplink VR traffic). For the downlink, timer-based flip detection may cause more retransmissions. Our test shows that, retransmission only triggers at most 1ms (1.9% out of 49.3ms reduction) more latency and 11.9 Kbps (0.1% out of 8.7 Mbps) more VR data transmissions. This also makes LTE-VR scale to VR devices similar to LTE.

10 PROJECTING LTE-VR TO 5G NETWORK

LTE-VR further sheds lights on the upcoming 5G design. We discuss how LTE-VR complements the 5G radio technologies (§10.1) and offers insights for 5G signaling protocol designs (§10.2).

10.1 Complementing 5G Radio Technologies

LTE-VR complements the recent efforts on 5G radio technologies. These efforts aim to expand wireless bandwidth (1000× proposed in [9, 35, 49]), and shorten the radio resource slots for smaller radio latency (0.2 ms, 4× smaller as proposed in [65, 73]). LTE-VR works directly with them for further latency reduction. Moreover, with new 5G radios, the transmission delay could be negligible even for high-quality VR (e.g., 4K 360 video). The signaling operations may dominate the network latency for mobile VR.

Solution	% of VR frames exceeding latency tolerance threshold		
	25 ms	15 ms	10 ms
1000× bandwidth, 0.2 ms time slot	0.081	0.92	2.58
The above change + LTE-VR	0.081	0.081	0.081
Improvements	N/A	11.36×	31.85×

Table 10. Estimation of high-quality mobile VR frames missing the human latency tolerance in 5G (1000× bandwidth).

We estimate how 5G radio affects LTE-VR's latency reduction. Assume the signaling design as 4G LTE, we emulate the 5G radio using 1000× bandwidth expansion and 0.2 ms slot. We rerun the evaluation of §9.1 for high-quality VR (60FPS, 4K). Table 10 shows that LTE-VR reduces the number of frames missing the human tolerance by up to 31.8×, even under the stringent 10ms latency threshold (e.g., [68]). As smaller VR latency is needed, LTE-VR becomes more beneficial since signaling dominates the network latency.

10.2 Hints for 5G Signaling Design

LTE-VR also provides insights for 5G signaling design.

Accelerate Error Recovery (§7.1.1). With faster 5G radios, the M-ACK→M-NACK flip may become more frequent with larger data volume. Current 5G proposals still retain the 4G control channel design *without redundant bits*. LTE-VR thus works with such proposals and offers a cost-effective approach to mitigating latency.

Mitigate HOL Blocking (§7.1.2). Existing 5G discussions [9, 12] retain the same RLC design, which is unfriendly to handover. LTE-VR thus still applies and offers hints to refine the RLC design.

Proactive Radio Resource Allocation (§7.2.1). Some 5G proposals seek to replace 4G LTE's uplink scheduling with random access or pre-allocation [65]. Note that the uplink traffic pattern for VR is inherently determined by the phone sensor (§4.1), so VR in 5G will still have periodic uplink packet. Our idea of predicting uplink VR packet arrivals still helps mitigate the latencies due to backoff in random access and resource waste in pre-allocation.

Mask Handover's Disruption via Prediction (§7.2.2). Some 5G proposals seek to replace hard handovers with soft ones [9]. While handover disruptions may be avoided, new network latency may arise. For example, out-of-order delivery may be amplified by accessing to multiple cells, thus prolonging delays at RLC. LTE-VR's handover prediction still helps VR take early adaptations.

11 DISCUSSION

Limitations. LTE-VR is not without its limitations. As a practical solution, it does not exploit the network-side opportunities to reduce latency. Moreover, its effectiveness partially relies on the mobile VR's regular uplink traffic patterns (from sensors). If this does not hold, LTE-VR still ensures latency no larger than the current LTE. Last but not least, LTE-VR focuses on mitigating the radio access latency. It is equally important to reduce the core network delay, which depends on edge server placement and congestion.

Applicability to other scenarios. LTE-VR also benefits other delay-sensitive apps, such as real-time mobile gaming, interactive augmented reality, autonomous driving, mobile healthcare, etc. These scenarios also require "anywhere, anytime" low-latency network access. LTE-VR can be extended to such scenarios, thus reducing their network latencies over LTE and future mobile networks. More concretely, all the above-mentioned applications will experience downlink flips, handovers, and HOL blocking in the mobile network. Our solutions provided in LTE-VR can thus mask or mitigate these latency components. In addition, some applications, such as mobile healthcare and augmented reality, collect and report extensive periodic sensor data. Our idea of pre-allocation of uplink resource can benefit these applications.

12 RELATED WORK

Improving the VR experiences is an active research topic. Extensive efforts are made to refine VR along multiple dimensions. VR apps can be optimized by local rendering [19, 41, 43, 54], video compression [24, 70] and traffic reduction [56]. At the OS level, research has focused on understanding and reducing the sensing and processing delays [38]. For the network, prior work has explored to increase wireless bandwidth [39, 58], leverage WiFi [13] or Bluetooth [33], or use the upcoming 5G [34, 35, 37, 64, 72]. Our work complements these efforts. We focus on the signaling operations, and study the feasibility of enabling mobile VR over 4G LTE and 5G.

Some previous works identify a few latency sources in LTE network [5, 7, 42]. Our work quantifies the severity of these latency components under VR traffic, and proposes solutions to mitigate them. Various techniques are proposed to reduce latency over LTE. They include application adaptations [15, 46, 69], traffic offloading [23, 40], adaptive congestion control [67, 71], efficient radio resource utilization [42, 57, 61], etc. Our work is orthogonal to these proposals. We quantify the impact of signaling operations on network latency. We also differ from another recent work [45], which reduces the LTE data *access* latency when establishing radio connectivity. We seek to reduce data *delivery* latency after access is granted.

13 CONCLUSION

Mobile VR promises to offer "anytime, anywhere" panorama view and immerse experiences in both static and mobile settings. A major challenge to enable mobile VR is to reduce its perceived latency. We show that, the data signaling operations contribute a large portion of network latency. Various latencies arise from both the interplays between protocols and the single-protocol deficiency. We thus devise LTE-VR, a client-side solution using side-channel information and cross-layer design.

In the broader scope, research on minimizing network latency for mobile VR warrants more efforts. While extensive studies have been reported to improve wireless access speed, they are not sufficient. It is equally important to optimize the signaling operations. We hope our study could stimulate more efforts in the research community to make mobile VR a first-class network citizen, and provide valuable input into the standardization of the upcoming 5G design and other apps such as augmented reality (AR).

Acknowledgments: We would like to thank our shepherd, Dr. Ranveer Chandra, and the anonymous reviewers for their constructive comments. This work is supported in part by NSF awards (ICN-WEN-1719403, CNS-1526456, CNS-1526985, CNS-1423576 and CNS-1421440).

REFERENCES

[1] 2011. ZTE UMTS Handover Prediction. (2011). http://www.authorstream.com/Presentation/sahaabhi-1282571-14-zte-umts-handover-description/.
[2] 2014. Huawei Handover Algorithm. (2014). https://www.slideshare.net/herobinh/08102014huawei-handovershandoveralgo.
[3] 3GPP. 2006. TS25.331: Radio Resource Control (RRC). (2006). http://www.3gpp.org/ftp/Specs/html-info/25331.htm
[4] 3GPP. 2012. TS36.322: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification. (Sep. 2012). http://www.3gpp.org/DynaReport/36322.htm
[5] 3GPP. 2014. TS36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification. (Mar. 2014). http://www.3gpp.org/DynaReport/36321.htm
[6] 3GPP. 2015. TS23.401: General Packet Radio Service enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access. (Dec. 2015). http://www.3gpp.org/ftp/Specs/html-info/23401.htm

[7] 3GPP. 2015. TS36.133: Requirements for support of radio resource management. (Mar. 2015). http://www.3gpp.org/ftp/Specs/html-info/36133.htm
[8] 3GPP. 2015. TS36.331: Radio Resource Control (RRC). (Mar. 2015). http://www.3gpp.org/ftp/Specs/html-info/36331.htm
[9] 3GPP. 2017. 3GPP 5G New Radio Working Group: Radio Interface architecture and protocols. (2017). http://www.3gpp.org/Specifications-groups/ran-plenary/46-ran2-radio-layer-2-and-radio-layer
[10] 3GPP. 2017. TS36.211: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation. (2017).
[11] 3GPP. 2017. TS36.213: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures. (2017).
[12] 3GPP. 2017. TS38.322: Technical Specification Group Radio Access Network; NR; Packet Data Convergence Protocol (PDCP) specification. (Jun. 2017).
[13] Omid Abari, Dinesh Bharadia, Austin Duffield, and Dina Katabi. 2016. Cutting the Cord in Virtual Reality. In *ACM HotNets*. 162–168.
[14] Michael Abrash. 2014. What VR Could, Should, and almost certainly Will be within two years. http://media.steampowered.com/apps/abrashblog/Abrash%20Dev%20Days%202014.pdf. (2014).
[15] Sharad Agarwal and Jacob R. Lorch. 2009. Matchmaking for Online Games and Other Latency-sensitive P2P Systems. In *ACM SIGCOMM*. 315–326.
[16] Robert S Allison, Laurence R Harris, Michael Jenkin, Urszula Jasiobedzka, and James E Zacher. 2001. Tolerance of temporal delay in virtual environments. In *IEEE Virtual Reality*. Yokohama, Japan, 247–254.
[17] ArsTechnica. 2016. Nvidia GTX 1080 review: The new performance king. https://arstechnica.com/gadgets/2016/05/nvidia-gtx-1080-review/4/. (May 2016).
[18] Lukas M Batteau, Alan Liu, JB Antoine Maintz, Yogendra Bhasin, and Mark W Bowyer. 2004. A study on the perception of haptics in surgical simulation. In *Springer Medical Simulation*. Berlin, Heidelberg, 185–192.
[19] Kevin Boos, David Chu, and Eduardo Cuervo. 2016. FlashBack: Immersive Virtual Reality on Mobile Devices via Rendering Memoization. In *ACM MobiSys*. 291–304.
[20] Hypergrid Business. 2016. Report: 98% of VR headsets sold in 2016 are for mobile phones. http://www.hypergridbusiness.com/2016/11/report-98-of-vr-headsets-sold-this-year-are-for-mobile-phones/. (Nov 2016).
[21] Tech Crunch. 2017. Google has shipped 10M Cardboard VR viewers, 160M Cardboard app downloads. https://techcrunch.com/2017/02/28/google-has-shipped-10m-cardboard-vr-viewers-160m-cardboard-app-downloads/. (Feb 2017).
[22] Android developers. 2017. Sensors Overview. https://developer.android.com/guide/topics/sensors/sensors_overview.html. (2017).
[23] Savio Dimatteo, Pan Hui, Bo Han, and Victor O. K. Li. 2011. Cellular Traffic Offloading through WiFi Networks.. In *IEEE MASS*. Valencia, Spain, 192–201.
[24] Charles Dunn and Brian Knott. 2017. Resolution-defined projections for virtual reality video compression. In *IEEE Virtual Reality*. Los Angeles, CA, USA, 337–338.
[25] Facebook. 2017. Facebook Spaces: A New Way To Connect With Friends In VR. https://newsroom.fb.com/news/2017/04/facebook-spaces/. (2017).
[26] GFXBench. 2017. 3D Graphics Performance of Google Pixel C. https://gfxbench.com/device.jsp?D=Google+Pixel+C. (July 2017).
[27] GFXBench. 2017. 3D Graphics Performance of Samsung Galaxy S7 edge (Mali-T880, SM-G935x). (March 2017).
[28] Google. 2017. Full OTA Images for Nexus and Pixel Devices. https://developers.google.com/android/ota. (2017).
[29] Google. 2017. Google Cardboard. https://vr.google.com/cardboard/. (2017).
[30] Google. 2017. Google Daydream. https://vr.google.com/daydream/. (2017).
[31] NVIDIA GRID. 2017. Virtualization Partners, NVIDIA GRID. http://www.nvidia.com/object/grid-partners.html. (2017).
[32] GTMoogle. 2016. StreamTheater: Hacking together MoonLight and Oculus Cinema for GearVR. https://github.com/GTMoogle/StreamTheater. (2016).
[33] Peter He. 2016. Virtual Reality for Budget Smartphones. *Young Scientists Journal* 18 (Jan 2016), 50–57.
[34] The White House. 2016. Administration Announces an Advanced Wireless Research Initiative, Building on President's Legacy of Forward-Leaning Broadband Policy. https://www.whitehouse.gov/the-press-office/2016/07/15/fact-sheet-administration-announces-advanced-wireless-research. (Jul. 2016).
[35] Huawei. 2013. 5G: A Technology Vision. http://www.huawei.com/5gwhitepaper/. (2013).
[36] Google Inc. 2017. Android.Telephony. (2017). http://developer.android.com/reference/android/telephony/package-summary.html.
[37] GSMA Intelligence. 2014. Understanding 5G: Perspectives on future technological advancements in mobile. https://www.gsmaintelligence.com/research/?file=141208-5g.pdf&download. (Dec. 2014).

[38] Teemu Kämäräinen, Matti Siekkinen, Antti Ylä-Jääski, Wenxiao Zhang, and Pan Hui. 2017. Dissecting the End-to-end Latency of Interactive Mobile Video Applications. In *ACM HotMobile*. New York, NY, USA, 61–66.

[39] Joongheon Kim, Jae-Jin Lee, and Woojoo Lee. 2017. Strategic Control of 60 GHz Millimeter-Wave High-Speed Wireless Links for Distributed Virtual Reality Platforms. *Mobile Information Systems* (2017), 10.

[40] Yoora Kim, Kyunghan Lee, and Ness B. Shroff. 2014. An Analytical Framework to Characterize the Efficiency and Delay in a Mobile Data Offloading System. In *ACM MobiHoc*. New York, NY, USA, 267–276.

[41] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, and Ningwei Dai. 2017. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. In *ACM Mobicom*. Snowbird, Utah, USA.

[42] Anna Larmo, Magnus Lindström, Michael Meyer, Ghyslain Pelletier, Johan Torsner, and Henning Wiemann. 2009. The LTE link-layer design. *IEEE Communications magazine* 47, 4 (2009).

[43] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn. 2015. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In *ACM Mobisys*. 14–17.

[44] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. 2016. MobileInsight: Extracting and Analyzing Cellular Network Information on Smartphones. In *The 22nd ACM Annual International Conference on Mobile Computing and Networking (Mobicom'16)*. New York, USA.

[45] Yuanjie Li, Zengwen Yuan, and Chunyi Peng. 2017. A Control-Plane Perspective on Reducing Data Access Latency in LTE Networks. In *ACM Mobicom*. Snowbird, Utah, USA.

[46] Justin Manweiler, Sharad Agarwal, Ming Zhang, Romit Roy Choudhury, and Paramvir Bahl. 2011. Switchboard: a matchmaking system for multiplayer mobile games. In *ACM MobiSys*. New York, NY, USA, 71–84.

[47] Microsoft. 2016. Does Microsoft have plan to add 4G or LTE to Hololens? https://forums.hololens.com/discussion/93/does-microsoft-have-plan-to-add-4g-or-lte-to-hololens. (2016).

[48] MobileInsight. 2017. MobileInsight. (May 2017). http://metro.cs.ucla.edu/mobile_insight.

[49] NGMN. 2015. NGMN 5G white paper. https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf. (2015).

[50] Navid Nikaein, Mahesh K Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. 2014. OpenAirInterface: A Flexible Platform for 5G Research. *ACM SIGCOMM Computer Communication Review* 44, 5 (2014), 33–38.

[51] NVIDIA. 2017. Nvidia GRID cloud gaming platform. http://www.nvidia.com/object/cloud-gaming.html. (2017).

[52] OpenAirInterface. 2017. OpenAirInterface Project. (2017). https://twiki.eurecom.fr/twiki/bin/view/OpenAirInterface.

[53] OpenSignal. 2016. The State of LTE (November 2016). https://opensignal.com/reports/2016/11/state-of-lte. (2016).

[54] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics* 35, 6 (2016), 179.

[55] Sony Playstation. 2017. PlayStation Now: Stream PS4 Games in 2017. http://blog.us.playstation.com/2017/03/13/playstation-now-stream-ps4-games-in-2017/. (2017).

[56] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 Video Delivery over Cellular Networks. In *ACM Workshop on All Things Cellular: Operations, Applications and Challenges*. New York, NY, USA, 1–6.

[57] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. 2010. Characterizing Radio Resource Allocation for 3G Networks. In *ACM IMC*. New York, NY, USA, 137–150.

[58] Qualcomm. 2016. Making immersive virtual reality possible in mobile. https://www.qualcomm.com/documents/making-immersive-virtual-reality-possible-mobile. (2016).

[59] Qualcomm. 2017. Qualcomm Chipchode Software: Radio Firmware Source. https://createpoint.qti.qualcomm.com/dashboard/public/productkit#public/product-kit/search. (2017).

[60] RiftCat. 2017. VRidge - Play PC VR on your Cardboard. https://riftcat.com/vridge. (2017).

[61] Sanae Rosen, Haokun Luo, Qi Alfred Chen, Z Morley Mao, Jie Hui, Aaron Drake, and Kevin Lau. 2014. Discovering fine-grained RRC state dynamics and performance impacts in cellular networks. In *ACM MobiCom*. New York, NY, USA, 177–188.

[62] Samsung. 2017. Samsung Gear VR. http://www.samsung.com/global/galaxy/gear-vr/. (2017).

[63] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell. 2011. Using Graphics Rendering Contexts to Enhance the Real-time Video Coding for Mobile Cloud Gaming. In *ACM MM*. New York, NY, USA, 103–112.

[64] SK Telecom. 2015. SK Telecom's 5G Architecture Design and Implementation Guidelines. http://www.sktelecom.com/img/pds/press/151020z_5G_architecture_design_and_implementation_guideline.PDF. (2015).

[65] Wen Tong. 2017. The Blueprint of 5G. In *IEEE International Conference on Communications*. IEEE.

[66] VRScout. 2017. Report: Global VR Hardware Revenue To Hit $3.6 Billion in 2017. https://vrscout.com/news/global-vr-hardware-revenue-forecaset/. (Feb 2017).

[67] Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan, et al. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks.. In *USENIX NSDI*. Lombard, IL, 459–471.
[68] NSF Workshop. 2017. NSF Follow-on Workshop on Ultra-Low Latency Wireless Networks. http://inlab.lab.asu.edu/nsf/files/WorkshopReport-2.pdf. (2017).
[69] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. 2015. pistream: Physical layer informed adaptive video streaming over lte. In *ACM MobiCom*. New York, NY, USA, 413–425.
[70] Matt Yu, Haricharan Lakshman, and Bernd Girod. 2015. Content adaptive representations of omnidirectional videos for cinematic virtual reality. In *ACM Workshop on Immersive Media Experiences*. New York, NY, USA, 1–6.
[71] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *ACM SIGCOMM*. New York, NY, USA, 509–522.
[72] Shunqing Zhang, Xiuqiang Xu, Yiqun Wu, and Lei Lu. 2014. 5G: Towards energy-efficient, low-latency and high-reliable communications networks. In *IEEE ICCS*. Macau, China, 197–201.
[73] Pingping Zong. 2016. 5G and Path to 5G. http://www.samsung.com/global/business/networks/events/Silicon-Valley-5G-Summit/attachments/S5_Intel_Pingping-Zong.pdf. (2016).

A METHODOLOGY

This appendix provides supplementary details for our experiments. We elaborate our experimental setup (A.1), the dataset we have collected (A.2), and the methodology of analyzing these data (A.3).

A.1 Experiment Setup

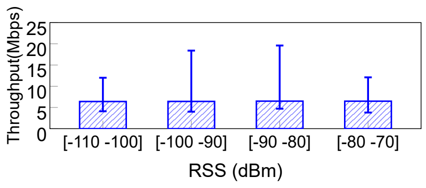
Device-side equipments and software: To collect the VR traffic, we use two representative mobile VR game platforms: StreamTheater [32] and VRidge [60]. We run both applications on multiple VR headsets, including the Gear VR (empowered by Samsung S7 Edge phones), and Google Cardboard/Daydream (both with Google Pixel and Huawei Nexus 6P). We play multiple games using StreamTheater/VRidge, as summarized in Table 2. When playing these games, we collect LTE's PHY/MAC/RLC/PDCP/RRC-layer messages to analyze the network latency. This is achieved by running MobileInsight [44] in the background together with the VR game.

To conduct experiments in large scale and collect extensive amount of the trace, we also implement an Android VR traffic generator to emulate VR uplink traffic on mobile phones, including Samsung S7 Edge, Huawei Nexus 6P, Google Pixel and Samsung S4. We implement a program to emulate the edge server behavior at the server side. We run two programs at the same time, and collectively they act similarly to a VR application and a cloud renderer. We tune the interval, packet size of the traffic generator, and throughput, interval of the server program to resemble the real VR application traffic pattern (as shown in Table 13).

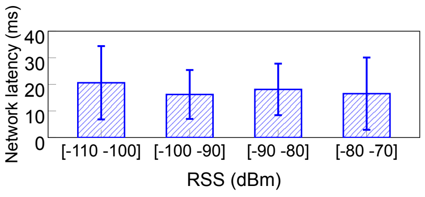
Edge server setup: For the edge server, we use a Dell XPS 8900 desktop with Nvidia GeForce GTX 745 GPU. The server is equipped with an Intel i7-6700 CPU and 16GB memory. The operating system running on the server is Windows 10. We deploy Nvidia SHEILD [51] software. The SHIELD streams the VR games from the computer to the mobile VR platform on the device (StreamTheater/VRidge), which further pushes the frame into the VR headset. Our scheme emulates edge processing that is being deployed at edge servers. For example, Nvidia partners with LTE vendors to deploy edge streaming service in the edge servers [31].

Experimental scenarios: We run three categories of tests when playing the mobile VR: static scenario (0m/s), slow mobility (walking, ~0.5m/s), and fast mobility (driving, ~50mph). We conduct 9-hour driving experiments in greater Los Angeles area, covering 870km distance. For driving experiments, we collect network traces on highway (~60mph) and local routes (~25mph). Usually each session takes 10 minutes. We conduct walking experiments on campus A with speed ~0.5m/s.

We do experiments in various radio signal strengths, ranging from -110dBm to -60dBm. The relationship between VR downlink throughput and RSS is shown in Figure 23a. The relation between network latency and RSS is shown in Figure 23b. They show that, the LTE could satisfy the VR downlink throughput (quantified in §4) in a large range of radio signal strength [-110, -70], even when it is relatively weak. However, LTE still cannot always satisfy the latency requirement even when within this range of signal strength (§5 – §6). Our findings in §5 - §6 hold regardless of radio signal strength, as they happen and could be found in our trace no matter what the signal strength is. Our solutions work irrespective of the signal strength as well, as they never depend on the radio environment. The average network latency along with its variance increases as the signal strength becomes worse. When the signal strength further degrades to ≤-110dBm, transmission time dominates the mobile VR's network latency. The mobile VR throughput requirement cannot be satisfied in this situation.



(a) Relationship between throughput and radio signal strength



(b) Relationship between latency and radio signal strength

Fig. 23. Radio signal strength versus throughput/latency.

A.2 Dataset

Table 11 summarizes our dataset. Our traces are collected in both weekdays and weekend, mainly during 7AM to 1AM. In total, we have 1094k uplink packets and 20126k downlink packets. We have 81 experiment sessions and each session lasts 3023 seconds on average. The total size of the trace is 33.89 GB. In total, we harvest 39.4k RRC messages, 2.2 million PDCP messages, 1.9 million RLC messages, 3.1 million MAC messages, and 2.7 million PHY messages. Besides, we also use user study traces under arbitrary traffic pattern from MobileInsight's open database [48], including 1.4 million RRC messages. Table 12 shows occurrences of handover on different device models. The handover happens without regard to the device.

A.3 Data Analysis Approaches

Network latency analysis: We extract the latency from each component as follows:

◦ *Prolonged Error Recovery (§5.1):* We use MAC layer and RLC layer network logs to validate that the flip exists, and calculate the RLC packet retransmission delay. MAC layer packet corruption is indicated with failed CRC (Cyclic Redundancy Check) check; when the downlink data is corrupted, we observe a failure in the CRC value in MAC downlink log. The device will then send an M-NACK in uplink, which could be observed in the uplink PHY PUCCH (Physical Uplink Control Channel) log. LTE base station uses multiple HARQ (Hybrid ARQ) processes to transmit data, each associated with a HARQ ID. Recovered MAC packets can be mapped with failed ones by checking the HARQ ID, since the base station HARQ process will hold for failed MAC packet transmission. Furthermore, the new data indicator (NDI) field stays the same, informing the device that this is a retransmission.

We now describe how we detect an M-NACK flip. After a MAC corruption, if the next packet comes with the same HARQ ID but a different NDI, we know the MAC retransmission is not triggered and the M-NACK is flipped. We can now analyze the RLC uplink log, where we first observe an R-NACK asking for the retransmission, and then the lost data is retransmitted in RLC

Condition	Device	Uplink packets #	Downlink packets #
Static	Galaxy Edge 7	37k	563k
	Google Pixel	36k	494k
	Huawei Nexus 6P	55k	822k
	Sumsung S4	25k	803k
Walking	Galaxy Edge 7	175k	2608k
	Google Pixel	67k	703k
	Huawei Nexus 6P	13k	160k
	Sumsung S4	51k	938k
Driving	Galaxy Edge 7	151k	2215k
	Google Pixel	764k	3106k
	Huawei Nexus 6P	679k	7647k
	Sumsung S4	41k	68k

Table 11. Dataset size

Galaxy Edge 7	Google Pixel	Huawei Nexus 6P	Samsung S4
155	211	565	13

Table 12. Handovers on different device models

Metrics	VR	Traffic Simulator
Downlink Throughput (Mbps)	8.9 ± 9.9	8.9 ± 7.2
Uplink Interval	17.4 ± 6.2	20.8 ± 1.0
Downlink Interval	15.0 ± 5.9	20.5 ± 0.5

Table 13. The closeness of emulator traffic to real VR traffic.

downlink. Using the timestamp of the RLC retransmission data and the original MAC downlink data, we calculate the prolonged error recovery latency.

∘ *Head-of-Line Blocking by Duplicates (§5.2):* The duplicate packets will be detected locally in the device chipset, as they appear the same PDCP sequence numbers with those that have been received before. By analyzing the PDCP layer data, we could calculate how many packets after the handover are duplicated, and how much latency they incur.

After the device experiences a handover, which we could learn from RRC layer log (RRC reconfiguration message with mobilityControl command), we observe the downlink PDCP data log. Each duplicate packet is marked as invalid by the device and then discarded. For these packets, we check the sequence number, and confirm in the previous PDCP log that they have indeed been received prior to the handover event, and thus are correctly detected. After the duplicate packets have been transmitted, we see in PDCP downlink log that new packets with valid new sequence number coming in. By combining the time of the first duplicate packet and the first new packet, we calculate the head-of-line blocking time.

∘ *Latency-Unfriendly Uplink Control Channel (§6.1):* We use physical layer network logs to get uplink delay breakdowns. With physical control channel PUCCH logs, we can get the knowledge of when the scheduling request is sent. By observing the MAC buffer change, we know when an uplink packet comes to the protocol stack and waits for the uplink grant. By mapping the SR outgoing time with buffer status, we calculate waiting delay T_{wait} as the lap between packet arrival time and SR sending time. The SR RTT T_{RTT} is calculated based on time when the SR is sent and

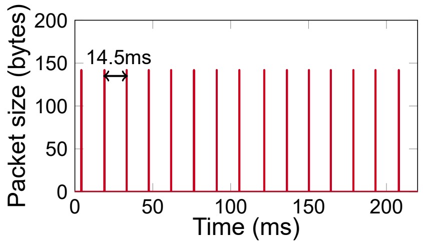


Fig. 24. Examples of VR traffic pattern (Airborne 1944 VR game in VRidge); Packets arrive at intervals 14.5 ± 0.4ms.

when the grant arrive, which can be learned from the downlink control channel PDCCH (Physical Downlink Control Channel) logs.

Processing delay T_{proc} is the interval of grant arrival and sending the first bit of the packet, which is available in the uplink MAC log. The uplink transmission delay $T_{UL-trans}$ is the lap between when the first bit of the packet is sent and when the last bit of the packet is sent. Both information can be learnt in uplink MAC buffer messages.

∘ *Long Disruptions with Hard Handover (§6.2):* In the LTE connected state, the base station will order the device when it should perform a handover. This information is available in the RRC layer log. After the handover, we could observe the device initiates a random access request to the new base station in the uplink PRACH (Physical Random-Access Channel). The base station responds to the device through PRACH, which means handover is done. After that, both uplink and downlink transmission are resumed. We observe the MAC uplink data and downlink data to get the timestamps of the first data after the handover. Using these information we are able to calculate the disruption time in both uplink and downlink.

Allocated bandwidth calculation: The uplink available grant is available in the downlink control channel in the collected trace. In order for the uplink data to send out, the base station will explicitly tell the device when and how much it can send the data. Therefore, we are able to gather this information at the device side in the uplink MAC layer log. By analyzing this information, we could calculate the allocated uplink throughput.

For downlink throughput, this information is readily to be calculated in the downlink data channel. We could analyze the related logs in the downlink PDCCH channel, where we can know the allocated bandwidth granted by the base station. We use this information to get the downlink real time bandwidth.

VR application uplink traffic pattern: Figure 24 plots a representative of VR uplink traffic pattern. We collect this trace when running Airborne 1944 VR game in VRidge. We observe similar pattern in other VR application uplink traffic. As we discuss in §4.1, VR application uplink packets are periodic and small. The uplink traffic carries motion data of similar size. The data is generated by the device built-in sensor, which samples the user motion periodically.

B LTE HANDOVER PREMIER

Overview. The old base station determines whether to perform the handover based on device-perceived radio qualities. Since the radio connectivity is established, the old base station sends the instructions to do measurement in command `Measurement_Configuration`. This configuration specifies two elements to the device (1) *Measurement object:* which cells to measure[9]; and (2) *Triggering events:* under which conditions a measurement report should be triggered (summarized in Table 14). Both elements are standardized in [8]. Whenever a candidate cell in the measurement

[9]Each base station can manage more than one cells, each covering a geographical area. We use "cells" and "base stations" interchangeably, for a slight abuse of notations.

Meas. Report Event	Explanation	Criteria
A1	Signal strength of the serving base station is getting better	$S_{serving} > h_{A1}$
A2	Signal strength of intra-frequency base station is getting worse	$S_{intra} < h_{A2}$
A3	Intra-frequency base station's signal strength is offset better than the serving base station	$S_{intra} > S_{serving} + h_{A3}$
A4	Intra-frequency base station's signal strength is getting better	$S_{intra} > h_{A4}$
A5	Inter-frequency base station's signal strength is good, while the serving base station's signal strength is bad	$S_{serving} < h^1_{A5}$ $S_{inter} > h^2_{A5}$

Table 14. 4G LTE's standardized criteria for device-side measurement report (standardized in [8]).

object meets the triggering condition, the device sends a `Measurement_Report`. Upon receiving these measurement reports, the base station will run its local decision algorithm to determine whether to perform the handover, and which cell the device should move to. If the handover should be performed, the old base station should first coordinate with the new base station. To retain the seamless service, it first creates a data tunnel to the new base station. Then it issues the `Handover_Command` to the device. Meanwhile, it tunnels the downlink data to the new base station, until the new base station receives the `handover_complete` command from the device. In this way, no data will be lost in handover.

Classifications of measurement report event. The standard defines different criteria to trigger the measurement report, as summarized in Table 14. Based on the 4G LTE cells to be measured, these criteria can be classified into three categories:

∘ *Serving base station measurement.* This refers to the measurements of the current serving base station. Event A1 will be triggered if the serving base station's radio signal strength is better than the threshold h_{A1}. It indicates the serving base station is good enough to provide radio communication. Instead, if the signal strength goes down below h_{A2}, Event A2 will be triggered. From the handover decision perspective, A2 is an alert of seeking new base station because the current channel condition is bad.

∘ *Intra-frequency base station measurement.* Intra-frequency cells refer to neighbor cells on the same frequency band as the serving cell. Since operated on same frequency band, the signal strength of the serving cell and neighboring cells are *directly comparable*. Event A3 and A4 belong to this category. They indicate whether a neighbor cell on the same frequency offers better radio quality than the serving cell. The base station defines the threshold h_{A3}, saying the report will be triggered if any intra-frequency neighbor's signal strength is h_{A3} offset better than the serving base station. This event is designed to look for potential handover target which can provide better service. Event A4 is to monitor if any intra-frequency neighbor's signal strength is better than threshold h_{A4}.

∘ *Inter-frequency base station measurement.* Inter-frequency cells refer to neighbor cells on different frequency band as the serving cell. Different from intra-frequency cells, inter-frequency cells' signal strengths are *not* directly comparable to the serving cell's since they are heterogeneous. Instead, the standard defines the indirect comparison using two thresholds. Event A5 is of this type. Signal strength of cells on different frequency bands are probably on different scales, so cannot be compared directly. To indicate that a inter-frequency cell has better channel condition, Event A5 has one threshold h^1_{A5} for serving cell and another threshold h^2_{A5} for inter-frequency cell. Report will be triggered only if two conditions are satisfied simultaneously. Event A5 is designed to seek inter-frequency base station with better channel conditions, therefore improving LTE performance.

Tolerating transient radio dynamics in measurement. The dynamics of signal strength can cause noisy measurement reports for the handover decision. To mitigate it, 4G LTE takes two approaches. First, it defines hysteresis in measurement report event (i.e. $h_{A1} - -h_{A5}$) to tolerate the transient dynamics. Second, a measurement window is defined in LTE (called time to trigger or TTT). LTE requires the signal strength should remain below or above the threshold for the time period of TTT. It effectively mitigates the report due to transient dynamics.

C ALTERNATIVES COMPARED WITH LTE-VR

In §9.1, we compare LTE-VR with three alternatives in evaluation: (1) **Legacy LTE:** This is the real operational 4G LTE, which serves as the baseline; (2) **Oracle LTE:** This is defined as LTE with global knowledge. The base station knows immediately when the device generates an uplink packet, when the M-ACK/NACK is flipped, and which packets have been received by the device before the handover. In our prototype, the base station accesses these information by reading the replayed logs. While not realizable in reality, it serves as the lower bound of the achievable latency in LTE; and (3) **LTE with bandwidth expansion:** It retains the same protocol design as LTE, but expands the wireless bandwidth. It is the well-known solution to reduce the latency.

Received November 2017, revised January 2018, accepted March 2018.