LDRP: Device-Centric Latency Diagnostic and Reduction for Cellular Networks without Root

Zhaowei Tan, Jinghao Zhao, Yuanjie Li, Yifei Xu, Yunqi Guo, Songwu Lu

Abstract— We design and implement LDRP, a device-based, standard-compliant solution to latency diagnosis and reduction in mobile networks without root privilege. LDRP takes a data-driven approach and works with a variety of latency-sensitive applications. After identifying elements in LTE uplink latency, we design LDRP that can infer the critical parameter used in data transmission and infer them for diagnosis. In addition, LDRP designates small dummy messages, which precede uplink data transmissions, thus eliminating latency elements due to power-saving, scheduling, etc. It imposes proper timing control among dummy messages and data packets to handle various conflicts. We achieve the latency diagnosis and reduction without requiring root privilege and ensure the latency is no worse than the legacy LTE design. The design of LDRP is also applicable for 5G . The evaluation shows that, LDRP infers the latency with at most 4% error and reduces the median LTE uplink latency by a factor up to 7.4× (from 42 to 5ms) for four apps over 4 mobile carriers.

1 Introduction

Low latency is critical to the proper functioning of various delay-sensitive mobile applications, such as mobile VR/AR, mobile gaming, mobile sensing, mobile machine learning, and emerging robot/drone-based image/speech recognition [1], [2], [3], [4]. These applications typically run on 4G LTE and 5G mobile networks, which offer ubiquitous access and seamless service. In this work, we study how to diagnose and reduce network latency over LTE networks for such applications in the connected state. This complements the work that reduces the connection setup latency [5].

Many emergent latency-sensitive mobile apps differentiate themselves for their heavier *uplink* data transfer (e.g., user motion control, sensory data, and live camera streaming) from the device to the infrastructure. Our experiments further reveal that, uplink latency contributes to a large portion of overall latency in tested apps over operational LTE. Diagnosing which latency elements are the bottleneck and reducing them is thus as important as reducing the downlink latency. While the downlink transfer has been extensively optimized, the uplink data transfer is less studied.

In this paper, we take one step further and aim to both diagnose and reduce uplink latency *on the application without root privilege or violating 3GPP standards*. This is challenging from two aspects. Firstly, the uplink data transfer in LTE is complex to diagnose and reduce due to complex interactions. This is because LTE adopts the feedback-based scheduling, on-demand radio resource allocation, retransmissions, etc. Second, these interactions are invisible to application layer without root. Mobile OS does not expose cellular-specific APIs to the applications.

We overcome the challenge with a key observation. The uplink latency for the applications with predictable packets can be diagnosed with cell-specific parameters. They mandate the latency for each device-base station interaction. In addition, these parameters can be inferred on the application layer. By sending packets in specific patterns and observing the latency differences on the application layer, we can infer critical parameters for each solution component. Both diagnosis and reduction operate on the device side, which complement those existing infrastructure-centric solutions that are better for downlink. Therefore, our solution is readily applicable to every off-the-shelf commodity device without root, including Android and iOS.

We thus design and implement LDRP, a device-based, application-layer, and software-only LTE latency diagnosis and reduction solution that is readily usable for *every* commodity smartphone device. To identify the elements for diagnosis, the overall design takes the data-driven approach. Through analysis of operational LTE traces, we identify all elements in LTE uplink latency, and quantify them via two popular applications. LDRP thus learns how each latency component can be inferred with critical LTE parameters. To enable its functions on every device, LDRP infers the parameters at the application layer without root privilege. It then uses the parameters to diagnose the latency elements.

LDRP further reduces the latency given the diagnosis results. The major goal is to reduce elements on the application layer with small overhead. No 3GPP standard change is needed. We find out that the transmission resources can be pre-acquired with early data packets. LDRP thus designates small dummy messages, which precede those uplink data packet transfers. It thus eliminates the latency elements due to power-saving and scheduling.

LDRP performs proper timing control among dummy messages and data packet streams to ensure *marginal data and power overhead*. It further resolves the conflict that arises among data packets and dummy messages. LDRP is also capable of predicting handover without root, thus providing applications with sufficient information to mask the latency. All these solution components only rely on the critical parameters learned for diagnosis without root privilege.

LDRP could be widely applied for applications in mobile networks. While LDRP is mainly designed for 4G LTE, it can be readily generalized to benefit the emergent 5G. LDRP mainly works for regular or predictable traffic, but can also help when the traffic is not strictly regular. LDRP does not

- *Z. Tan, J. Zhao, Y. Xu, Y. Guo, and S. Lu are with University of California, Los Angeles. Email: {tan, jzhao, yxu, luckiday, slu}@cs.ucla.edu*
- *Y. Li is with Tsinghua University. Email: yuanjiel@tsinghua.edu.cn*

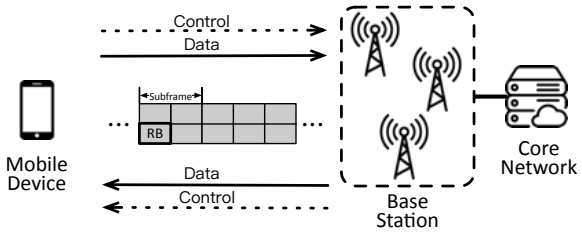


Fig. 1: Data transmission over Mobile Networks.

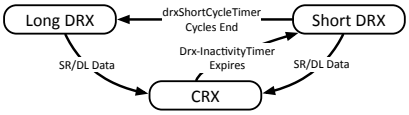


Fig. 2: State transition for LTE DRX power-saving.

arise fairness concern, as the sent dummy packets are indeed delivered and charged by networks, without breaking any standardized operation. It has no impact on network resource efficiency or other application's performance. LDRP leverages the practice that a base station usually schedules more resource than requested data, which could be used to carry the additional dummy packet.

We implement LDRP on commodity Android phones (§7) and our experiments confirm its effectiveness (§8). LDRP can diagnose each latency component with at most 4% error. It then reduces the median LTE uplink latency by up to 7.4× (from 42ms to 5ms) for four tested applications over four US mobile carriers. In any case, LDRP ensures that the network latency for data transfer is no worse than the legacy LTE design. The energy and data volume overhead is negligible.

2 Background

2.1 Mobile Networks Primer

Architecture The mobile networks, such as 4G LTE and 5G, offer the only large-scale infrastructure that ensures universal coverage and "anytime, anywhere" access. Its infrastructure consists of radio base stations (BSes) and the core network (see Figure 1). A mobile device transfers its data with a local BS ("cell"), which covers a geographic area. The BS further relays the data to the Internet via the core.

Data Transmission A mobile device has uplink (or UL, device→BS) and downlink (or DL, BS→device) transmissions. In 4G/5G, data transfer uses scheduling-based mechanisms. Each data must be delivered in the granted time and frequency units, called Resource Block (RB). For uplink grant, a device sends a Scheduling Request (SR) to the BS in Physical Uplink Control Channel (PUCCH). Upon receiving it, the grant is sent to the device via Downlink Control Indicator (DCI) in Physical Downlink Control Channel (PDCCH). The device can then send user data in the scheduled RB in Physical Uplink Shared Channel (PUSCH). For downlink, BS directly allocates RBs upon data arrival in PDCCH without extra requests. The downlink data is sent in Physical Downlink Shared Channel (PDSCH).

Power Saving through DRX The power-saving mechanism Discontinuous Reception (DRX) is a technique for a device to save power over LTE. Instead of continuously waking up for potential downlink delivery from the BS, the device might sleep in the absence of data transfer, thus reducing its energy consumption. In DRX, a device has three states: Long DRX Cycle, Short DRX Cycle, and Continuous Reception (CRX) [6]. In CRX, the device wakes up during the ON period to monitor downlink channels. In long/short DRX, the device only wakes up for a short period of time (set by the onDuration Timer) at the start of each DRX cycle. It dozes off during the OFF period for the remaining time.

The DRX state transition is shown in Figure 2. In the Long/Short cycle state, if any downlink data is received during the ON period, the device enters the CRX state and starts the drx-InactivityTimer. Upon sending uplink data, the device initiates an SR request. It then switches to the CRX state as well. If the device receives downlink data or initiates another SR request, the timer restarts. The short DRX state is entered once the drx-InactivityTimer expires. In this state, the device enters long DRX after the number of drxShortCycleTimer short cycles. All such involved timer parameters are negotiated between the device and the BS during connection setup through RRC.

Handover in LTE A device might disconnect from the serving BS and connect to a new one for better signal. This is called handover (or HO). The serving BS determines whether to perform the handover based on device-perceived radio qualities due to mobility. After the radio connectivity is established, the serving BS sends the instructions for the device to perform measurement on neighboring cells. When the measurement results meet the pre-configured conditions, a report is sent to the serving BS. It will determine the handover decision based on the results and send command to the device. The device first disconnects from the old BS, and then connects to the new BS.

2.2 Latency-Sensitive Apps over Mobile Networks

In this section, we exemplify some representative latency-sensitive applications over the mobile networks.

Mobile VR A mobile virtual reality (VR) app typically involves 3D scenes and associated graphical engines [1], [7], [8], [9]. Standalone VR headsets such as Google Daydream [10] render 3D scenes locally. However, due to limited computation resources and high power consumption on mobile devices, high-quality VR applications typically need the edge/cloud servers to offload the rendering task [11]. In this client-server scheme, the mobile headsets or pads provide sensory/control data, while the server renders the 3D scene in the form of graphical frames. The server coordinates multiple devices, renders the VR graphical frames based on the device's input, and constructs the appropriate 3D scene for each given device.

• *Showcase VR prototype:* Following the above paradigm, we have built an example VR game with Unity 3D engine [12] on Android phones to study latency in mobile networks. It has three modules: the controller at the device, the camera controller at the server, and the streaming component. The Android controller app acquires the device rotation data from the gyroscope sensor to control the in-game camera rotation. The GPS location is fed into the VR game so that the virtual character moves with the player's location updates. Upon receiving the player's sensory data, the camera controller at the server processes them and makes corresponding position and rotation movements for the virtual

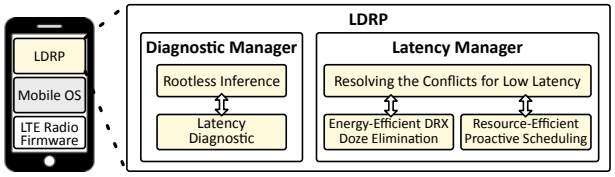


Fig. 3: An overview of LDRP.

camera. We implement the streaming module with open-sourced libraries Unity Render Streaming [13] and WebRTC for Unity [14]. With the streaming module, the camera view is rendered and streamed back in 60FPS to the player with WebRTC. Players open the camera stream with the Web browser on the phone to get the real-time camera view.

Mobile sensing Smartphones today are equipped with multiple sensors: accelerometer, gyroscope, camera, microphone, to name a few. Many mobile sensing apps collect sensory data and upload them at runtime to the cloud for fast processing. For example, a localization app sends the GPS data to the cloud for real-time navigation. All such sensing apps are latency sensitive.

Mobile gaming In multi-player mobile games, the device acts as a controller that collects user motion, while the remote server processes the game logic. The server further provides proper synchronization and coordination among players. Moreover, pure cloud-based gaming (with rendering being processed in the cloud) is also trendy [1]. It is a new gaming paradigm being pushed by companies [2].

Cloud/edge-assisted machine learning Mobile apps with machine learning features (e.g., image/object recognition or speech understanding [3], [15]) also pose latency requirements. Network latency becomes a bottleneck for smart assistants, such as Alexa [16] and Siri [17]. Users may tolerate at most 200ms response time, while deep learning based local transcriptions take only 10ms [18].

Networking usage patterns by these mobile apps All the above representative mobile apps involve *frequent and regular uplink* data transfer. The mobile VR, sensing, and gaming [19], [20] applications collect data from device sensors and upload them to the server for subsequent actions. These sensors typically produce small data *periodically*. The user can only configure the sampling periodicity through the API provided by the mobile OS [21]. The machine learning based apps also have predictable traffic. They typically perform local computations with predictable latency before an uplink data transfer. For example, face recognition apps process a video frame locally using a fixed-sized neural network (NN). A user can gauge the delay based on the NN size. Emerging robotic or drone-based applications perform local tasks for a certain duration (e.g., scanning the surrounding environment for a few seconds [22]) before uploading the result. Such apps also exhibit uplink traffic that can be accurately predicted.

3 LDRP Overview

We devise LDRP, an in-device software solution to latency for mobile apps. Figure 3 shows LDRP's components. It has two major functionalities, Latency Diagnosis and Latency Reducer, which provide the user apps with real-time breakdown analysis and reduction of LTE network latency, respectively. LDRP runs as a user-space daemon and is thus applicable to both Android and iOS. Its benefits come *w*ithout system/root privilege, firmware modification, or hardware support. To achieve so, both components infer critical cellular parameters by carefully-constructed schemes at the application layer without root access.

- **Latency Diagnosis (§4):** LDRP diagnoses latency elements in LTE network for applications. By a data-driven analysis on LTE latency analysis, we find that uplink latency is the bottleneck and each latency element has a strong correlation with cell-specific configurations. Therefore, LDRP leverages these parameters to infer each latency element for diagnosis. Although these low-level parameters are not accessible without root, Latency Diagnosis designs intelligent algorithms to accurately infer them in the application layer *without root*.
- **Latency Reducer (§5):** LDRP masks the LTE latency elements on the application layer with inferred parameters. As an application-layer solution, LDRP cannot directly control the low-level LTE mechanisms (that require root privilege or even firmware change). Instead, it indirectly regulates the LTE uplink transfer with well-crafted *dummy packets*. They *proactively* request the needed radio resources and high-speed transfer mode with standard-compliant mechanisms. By selecting proper timing based on parameters, the solution retains low energy and data consumption overhead. LDRP complements solutions designed to reduce other non-network latency elements [23], [24], [25].

4 Rootless Latency Diagnostics

In this section, we introduce how LDRP diagnoses the latency from the application layer. For this purpose, we address three issues:

- *How large is the uplink latency over LTE?* We use measurements to quantify it in §4.1.
- *Why is the uplink latency prohibitively high over LTE?* We break down this latency into multiple elements. We quantify their impact, identify root causes, and share insights in §4.2. We show that, each element is closely related to a few LTE parameters assigned by the BS. With these parameters, the latency can be deduced.
- *How to infer these parameters from the application layer?* We further show that in §4.3, these LTE-specific parameters can be inferred from the application layer, without any infrastructure assistance or root privilege. Therefore, LDRP infers these parameters and leverages them to diagnose the latency components.

4.1 Measuring LTE Latency: Who is the Bottleneck

Methodology We analyze the traces from our showcase VR game and another popular mobile application PUBG Mobile [2]. Our VR application uploads user motion packets (∼60Bytes) and receives 60FPS, 5Mbps downlink video

1. We cannot measure VR downlink network latency on AT&T and Sprint, since their firewalls in core networks block the traffic. However, we could still study the uplink latency breakdown between device and BS (e.g., in Table 2).

App	Latency	AT&T	T-Mobile	Verizon	Sprint
PUBG	UL Net	10.7	9.9	10.0	17.7
	DL Net	5.0	5.0	5.0	5.0
	UL/Total	68.2%	66.4%	66.7%	78.0%
VR	UL Net	N/A [1]	18.4	23.8	N/A
	DL Net	N/A	8.5	10.6	N/A
	UL/Total	N/A	68.4%	69.2%	N/A

TABLE 1: LTE latency (ms) for two mobile apps.

stream. Downlink packets are sent from the server to the device over LTE. PUBG is a mobile game with frequent uplink data (∼40ms interval) and downlink responses. Both uplink and downlink packets are small (<100Bytes). The latency due to server processing is less than 1ms. The mobile devices (a Pixel 2 and a Pixel XL) run the apps. We collect both app logs and LTE signaling traces via MobileInsight [26]. We carry out our experiments over four US mobile carriers from 12/2019 to 09/2020. The tests cover static, low-mobility (∼1m/s), and high-mobility (∼30mph) cases, with varying signal strength (-120∼-80dBm).

Results We measure the LTE uplink latency. We monitor the device buffer and compute the latency for each data packet. This information is available in the MobileInsight message "LTE MAC UL Buffer Status Internal". Despite small packet size, the uplink latency turns out to be non-negligible, as shown in Table 1. For all four carriers, the uplink latency (UL NET) ranges from 9.9-17.7ms for PUBG and 18.4-23.8ms for VR. These latency values might not meet the requirements of a number of latency-sensitive apps [27].

Who is the latency bottleneck? We further discover that, instead of downlink, the uplink latency poses as a major component in overall latency. We compute the downlink latency from logs of "MAC DL Transport Block" in MobileInsight. The results (DL NET) are in Table 1. We see that, uplink latency accounts for 66.4-78.0% in PUBG and 68.4-69.2% in VR. Surprisingly, even for the downlink-heavy VR app, uplink latency still contributes to a large portion of the overall latency. Recent techniques (e.g., MIMO and carrier aggregation) and 5G further reduce the DL latency with faster PHY designs. In contrast, as we will see later, the scheduling design employed for the uplink will likely be retained in 5G. As a result, we will focus on the uplink latency in this paper.

Disruption during Handover We also analyze the latency triggered by handover. Due to the PHY layer design in LTE, any uplink or downlink service will be interrupted during the switching. We find the handover event in the message "RRC OTA PACKET" and then check the disruption latency from "MAC UL TRANSPORT BLOCK".

4.2 Uplink Latency Components for Diagnostics

We next analyze the root causes for long network latency in 4G LTE. We identify various latency elements for the LTE uplink latency by analyzing the 3GPP standards [6], [28].

We breakdown the uplink latency as shown in Figure 4. The average number of each latency element is shown in Table 2. We can observe that, the major uplink latency bottlenecks are T_{drx_doze}, T_{sr_grant}, and T_{sr_wait}, while T_{bsr_grant} and T_{retx} are one magnitude smaller compared to other elements. When a handover happens, T_{HO} will disable the UL access for more than 50ms in every operator.

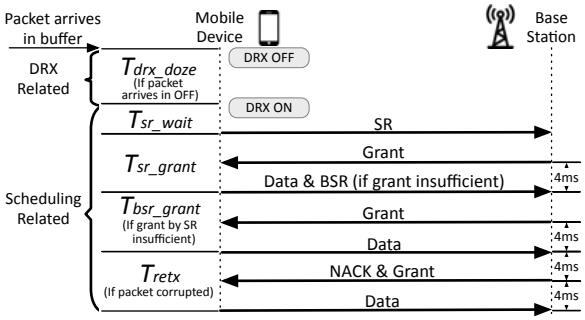


Fig. 4: LTE uplink procedure & latency elements.

Latency (ms)	AT&T	T-Mobile	Verizon	Sprint
T_{drx_doze}	29.7	31.9	28.3	29.2
T_{sr_wait}	4.4	4.4	4.6	9.0
T_{sr_grant}	8.2	8.5	8.0	10.1
T_{bsr_grant}	0.03	0.00	0.03	0.16
T_{retx}	0.17	0.14	0.32	0.72
T_{HO}	59.4	60.6	75.7	52.6

TABLE 2: Measured UL latency elements for VR application. T_{drx_doze} is the average value when present.

4.2.1 DRX Doze Latency.

How downlink DRX incurs long uplink latency DRX is designated for power saving over *downlink* transmissions. It should not block any uplink transfer. In fact, the 3GPP specification [6] stipulates that, upon the uplink sending an SR, downlink DRX should enter the CRX state as if receiving a downlink data packet. However, we found that this is not the case in practice. A new data packet refuses to invoke an SR if the device is in the doze mode. Instead, it continues to doze for a while (the time is denoted as T_{drx_doze}). It then waits for an SR slot to initiate the SR, while migrating the device to the CRX state. Table 2 shows that, T_{drx_doze} is 28.3-31.9ms on average in the four carriers. The maximum latency is 59ms with the 90th percentile being 42ms.

Note that the DRX doze latency is different from the known downlink delay due to waiting for DRX ON state. 3GPP [6], [28] does not mandate to prepare for SR at the DRX state. Although this latency element is not standardized, it is common for vendors as they use DRX doze to save energy. The DRX-induced doze timer is hinted in Qualcomm patents [29], where the device defers its SR during DRX OFF for energy savings. We indirectly validate this behavior in a ZTE Z820 with Mediatek Chipset. For packets with an interval of 1 second, the measured average RTT is 35ms longer than that of packets with a small interval.

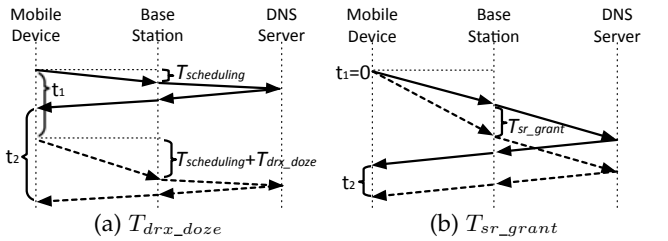
4.2.2 Scheduling Latency.

How scheduling incurs long latency We show in §2 that uplink data *cannot* be immediately sent out before the device is granted resource. An SR must be sent through PUCCH for uplink grants. However, an SR signaling cannot be sent at any time. It can only be sent during certain time slots (called SR occasions). The periodicity of SR occasions is notified by the BS during connection setup.

Therefore, the device must wait for an SR occasion before receiving a grant from the BS to upload its data packet. The latency element, denoted as T_{sr_wait}, is thus affected by the periodicity of an SR occasion T_{sr_period}. The device

Parameters		AT&T	T-Mobile	Verizon	Sprint
T_{sr_grant}	8ms	96.6%	96.5%	98.8%	0
	10ms	0	0.2%	0.1%	98.1%
	others	3.4%	3.3%	1.2%	1.9%
T_{sr_period}	10ms	94.0%	98.1%	92.3%	11.9%
	20ms	6.0%	1.9%	0	48.9%
	40ms	0	0	7.7%	39.%
$T_{inactivity}$	200ms	100.0%	99.5%	99.6%	84.5%
	others	0	0.5%	0.4%	15.5%

TABLE 3: Critical LTE parameters for uplink latency.



(a) T_{drx_doze} (b) T_{sr_grant}

Fig. 5: Inferring parameters at application layer.

then waits for a grant, which the device could use 4ms later. If the grant is not received before the next available slot due to SR failure, the device will resend SR until a grant is received. The latency from sending the first SR to sending the data packet is denoted as T_{sr_grant}. The two elements of scheduling are shown in Figure 4. We measure them in Table 2. The SR waiting latency T_{sr_wait} is 4.4ms for AT&T, 4.4ms for T-Mobile, 4.6ms for Verizon, and 9.0ms for Sprint. Sprint has the largest T_{sr_wait} because it has the longest SR cycle. T_{sr_grant} is 8.2ms, 8.5ms, 8.0ms, and 10.1ms for the four carriers. The accumulative latency is denoted as $T_{scheduling} = T_{sr\_wait} + T_{sr\_grant}$.

4.2.3 Handover Disruption

Another latency deficiency stems from handover (HO). Table 2 shows that, the average disruptions are 59.4 ms, 60.6 ms, 75.7 ms, and 52.6 ms in AT&T, T-Mobile, Verizon, and Sprint, respectively.

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 [30] and the device maintains concurrent access 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 connectivity to both base stations. 4G thus decides to not support soft handover.

Long latency for LTE hard HO. Following the standards [31], we derive the bound of the disruption time in LTE handover as follows $T_{HO} \leq T_{device-proc} + T_{random-access}$ where $T_{device-proc}$ is the time spent by the device to prepare for connecting to the new BS, and $T_{random-access}$ is the random access round-trip. $T_{device-proc}$ can be further decomposed as $T_{search} + 20\ ms$, where T_{search} is the scanning of the new BS. To guarantee sufficient time for local processing, [31] allows for 20ms safeguard interval. In reality, as we observed in all operators, the HO disruption time is much larger than 20ms, as the HO preparation and switching logic for new cell will consume no less than 30ms. Such design choice of hard handover is thus not friendly for latency-sensitive applications.

4.2.4 Minor latency elements not in diagnosis.

Buffer status report (BSR) SR is an indicator that informs the BS of new pending data, without specifying *how much*. When the packet that triggers SR is large, the initial grant might be insufficient. The device then sends a BSR (Buffer Status Report) together with the data packets in the scheduled RBs. Unlike SR, a BSR includes the info on how much data remains in the device buffer. Upon receiving the BSR, the BS will process it and respond with sufficient grants for the buffered uplink data.

We note BSR's impact on uplink latency is negligible for most applications in §2.2. The latency between a BSR and the time to use the grant (denoted as T_{bsr_grant}) is illustrated in Figure 4. Conceptually, it is the request processing time + 4ms, similarly to T_{sr_grant} ($\approx 10ms$). However, it equals to 0 when the initial grant is sufficient. The measurement results are in Table 2. The BSR latency is less than 1ms on average. This is because a BS usually provides a large grant (>100B) sufficient for our apps in response to SR.

MAC Retransmission latency An uplink data packet might be corrupted during transfer. Upon receiving a corrupted packet, the BS notifies the device by sending a NACK and a grant. The device uses the grant to retransmit the corrupted data. Similar to BSR latency, the retransmission has limited impact on the uplink latency for apps in §2.2. The ReTx latency for uplink data packet is fixed at 8ms if needed [28] and 0 otherwise. We denote this latency as T_{retx} and the procedure is shown in Figure 4. Among all data packets, 2.1% in AT&T, 1.7% in T-Mobile, 4.0% in Verizon, and 9.1% in Sprint perceive ReTx latency. Less than 1ms latency is incurred on average, shown in Table 2. Unlike downlink with up to 10% retransmissions [32], uplink packets are small and less prone to corruption.

4.3 Rootless Diagnostics

We note that, the uplink latency decomposition can be calculated by observing each operation in the mobile networks. Unfortunately, observing such events from the application is impossible without root. The existing tools all require system privilege (e.g., MobileInsight [26]) or additional hardware (e.g., QXDM [33]), since they require the mobile OS to open Diag port for cellular-specific information. Our goal is to let `LDRP` work with *every* commodity device; therefore, we seek to infer these parameters or events at the application layer.

From the previous analysis, we learn a critical insight. The latency components depend on LTE parameters or remain near-constant under a base station. To infer these critical LTE configurations, `LDRP` takes the following idea. If we can send packets with certain pattern and infer which latency element is experienced by different packets, we can observe their latency difference to infer each parameter.

With this idea, `LDRP` exploits packet pairs for probing. Figure 5 shows the general procedure. `LDRP` sends two consecutive probing requests and records their interval t_1. Upon receiving the responses to both packets, `LDRP` compares the responses' intervals t_2 with t_1, and estimates the corresponding timers. This approach is based on the

premise that, the difference between t_1 and t_2 mainly arises from the different uplink LTE latency experienced by two packets. This premise largely holds in practice, because latency fluctuations from the base station are much larger than those in the core network or servers[2]. Compared with the conventional packet-pair technique, LDRP customizes probing packets with the LTE domain knowledge for accurate inference.

Diagnose DRX Doze Latency Recall that the DRX doze latency is only present when the packet interval is large. The idea is to let t_1 be large enough so that the first response packet cannot keep the second request in DRX ON. The second packet in the pair experiences UL DRX doze latency, while the first does not as we immediately start next pair after one is done. We can thus use the interval difference $t_2 - t_1$ to infer DRX doze latency. Considering that two requests can also be different in terms of scheduling latency, we repeat the pair for 10 times and take the interval difference average. Figure 5(a) illustrates this procedure.

One caveat is that we need to know how large t_1 is so that the second request suffers from DRX doze latency. We increase the interval t_1 gradually until a certain spike appears in measured RTT for the second request, caused by DRX doze latency (≈30ms as shown in §4). The time interval between the first response and the second request that triggers such spike infers $T_{inactivity}$. We can thus diagnose that $T_{drx\_doze} = t_2 - t_1$.

Diagnose Scheduling Latency To diagnose the scheduling latency, LDRP needs to infer T_{sr_grant} and T_{sr_wait}. Figure 5(b) shows how LDRP learns T_{sr_grant}. We let t_1 as 0 by sending both requests together. As we just showed, the grant is sufficient for a single request packet. We increase the size of the second request so that the grant will not be sufficient for both request packets. According to the scheme, the first request experiences only scheduling latency while the second experiences the same scheduling latency plus T_{bsr_grant}, which equals to T_{sr_grant} under a same BS. We thus can derive latency element T_{sr_grant} from the measured t_2 as $T_{sr\_grant} = t_2$. Besides, T_{sr_wait} is the wait time for scheduling request, expected to be $1/2 \cdot T_{sr\_period} - 1$. To get T_{sr_period}, we enumerate all possible SR periodicity in [28], from the largest to the smallest. For each periodicity T_p^i, LDRP sends 15 packet pairs with $t_1 = T_p^i$. If $T_{sr\_period} < T_p^i$, all pairs will be sent out separately with different grants, so $t_2 > 0$. For a T_p^i that LDRP spots a pair with $t_2 = 0$, we find $T_{sr\_period} = T_p^i$. With both inferred from application without root, LDRP diagnoses $T_{scheduling} = T_{sr\_wait} + T_{sr\_grant}$.

Diagnose Handover Latency To diagnose handover latency, LDRP does not require packet pair. Instead, it uses the OS API to query the current cell and monitor the cell change. Once the cell is switched after handover, LDRP records the time of the last packet before the cell change and the first packet after the handover. This is the inferred handover disruption time experienced by the device.

2. We have validated this premise in operational LTE. We send a pair of DNS requests at $t_1 = 0$. A UL grant suffices to send both requests; they arrive at the BS simultaneously. t_2 is solely affected by the core network and DL. The results show that $t_2 < 1ms$ for >99% responses.

5 LDRP Rootless Latency Reduction

The next goal of LDRP is to reduce latency elements after quantifying them. Similar to diagnosis, we aim to achieve latency reduction without *any privilege*, such as root or firmware access. Therefore, LDRP must be standard-compliant, while naively changing current protocols or scheduling practice (e.g., keep sending an SR) is out of the question.

The design is based on one insight from our diagnosis. For doze and scheduling latency elements, they can be proactively reduced with small dummy packets. These packets can wake up the device or request resources before the data arrival. However, this is not trivial. Sending these packets improperly can incur huge power or even hurt latency. Therefore, they must be timed with the critical configurations from the diagnosis.

We next elaborate on how LDRP leverages this insight to mitigate the uplink latency while achieving the most latency reduction with low overhead. We also show that, handover could be predicted with limited information exposed to the application layer. LDRP addresses three challenges:

- **Accurate dummy packet delivery for each latency element to maximize reduction and minimize cost (§5.1–5.2)** : Initializing the dummy packets at the right time is crucial to both reducing latency and minimizing energy consumption, signaling overhead, and radio resource usage. The proper timing depends on critical parameters for each latency element and traffic pattern. To this end, LDRP customizes the timing control for critical latency elements, including the DRX doze and scheduling (§4.2).
- **Conflict handling for overall latency reduction (§5.3):** Simply reducing each latency element does not suffice to reduce the overall latency. Due to the complex interactions between LTE latency elements, reducing one latency element may increase other latency elements. Moreover, the dummy packets may compete radio resources with the legitimate data, incurring additional data latency. To this end, LDRP devises resolution and avoidance schemes for both types of conflicts.
- **Handover Prediction (§5.4):** Since the HO hard disruption is rooted in LTE design, it is critical to predict it and notify the application for preparation. Henceforth, LDRP provides a lightweight application-layer predictor which accurately predicts the HO occurrence with reasonable earliness. This is achieved by using measurement reports and handover stateful design.

5.1 Energy-Efficient DRX Doze Elimination

To reduce the DRX doze latency in §4.2.1, LDRP should ensure the device is in ON period when a data packet arrives at the device buffer. As an application layer solution, LDRP cannot directly switch the device to the CRX state (that needs firmware modification). Instead, it sends a dummy packet (*rouser*) *before* the data packet's arrival.

Seemingly straightforward, *rouser* is only effective if sent at the right time. An imprudent *rouser* can either incur unacceptable energy waste or cannot help reduce latency. Therefore, timing control is crucial to balancing latency and energy cost. We first discuss some straightforward solutions with limitations, and then present our design.

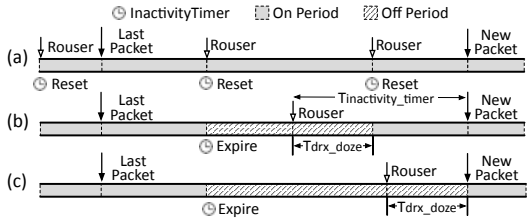


Fig. 6: Component solution to DRX doze latency.

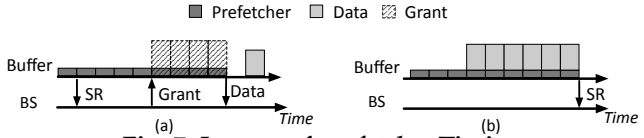


Fig. 7: Impact of *prefetcher* Timing.

Naive timing control One naive solution is to keep DRX at CRX state at all times by frequently sending *rouser*s. As shown in Figure 6(a), this can be achieved by sending a *rouser* every $T_{inactivity}$. Unfortunately, this results in unacceptable energy waste, as the device never enters DRX OFF.

A better choice is to send a *rouser* with the time in advance, denoted as t_r, being set to $t_r = T_{inactivity}$ (Figure 6(b)). On one hand, as the packet keeps the ON period for $T_{inactivity}$ after dozing, $t_r = T_{inactivity}$ ensures that the data packet enters the buffer during the ON period. On the other hand, this saves power compared to the first naive choice, since the extra ON period is capped at $T_{inactivity}$ for each packet at most. However, extra energy consumption is still incurred. Since $T_{inactivity}$ (∼200ms) is typically much larger than T_{drx_doze} (∼30ms) in reality, the ON period between wakeup from the doze mode and the data packet is unnecessary.

LDRP's approach LDRP prioritizes latency over marginal energy waste with proper timing control. Instead of frequent *rouser*s in naive solutions, LDRP only sends a *rouser* for the time T_{drx_doze} in advance. If the device enters the ON period during doze, i.e. $t_r > T_{drx\_doze}$, the *rouser* finishes dozing before the data packet arrives, thus eliminating the doze latency for the data packet. It is also likely that T_{drx_doze} for a *rouser* exceeds t_r. In this case, the packet enters the buffer and endures the dozing latency together with the *rouser*. Although the doze latency is not eliminated, the *rouser* reduces it by t_r.

The solution can be achieved in the application without root. First, the required T_{drx_doze} parameter can be inferred with techniques introduced in §4.3. We note that, the inference on the application layer could be smaller than the actual T_{drx_doze} due to the packets arriving in DRX ON. Therefore, we run the inference for 3 times and select the maximum reference result of T_{drx_doze} denoted as $T_{drx\_doze\_max}$ to prioritize latency reduction. Second, sending *rouser* can also be done by a normal application. No infrastructure or extra privilege is used in the process.

5.2 Resource-Efficient Proactive Scheduling

LDRP next seeks to mask the round trips of the scheduling in §4.2.2 for the mobile app. The idea is to send a scheduling request (SR) *before* the arrival of the data, so that the data does not need to wait for the radio grants. As an application-layer solution, LDRP cannot directly trigger the SR early (which requires modifying the firmware). Instead,

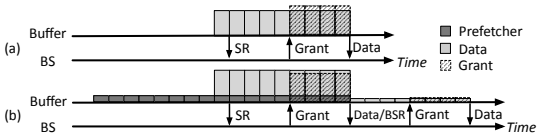


Fig. 8: Corner case: a *prefetcher* increases latency.

it requests a grant from the BS in advance by sending a dummy message, named *prefetcher*. This is feasible since the grant is not tied with the packet that requests it. Moreover, since the BS responds to each SR regardless of the pending data size, a small dummy message can receive a grant that allows for much-larger-size transmission than itself, thus sufficing to accommodate the followup data packet transfer in a single transmission.

Similar to the DRX doze elimination in §5.1, an effective *prefetcher* also needs accurate timing control. As shown in Figure 7, imprudent timing can offset the latency reduction, and/or waste radio resources. We next discuss both naive solutions in Figure 7, and then show LDRP's approach.

Naive timing control An early *prefetcher* might result in both resource waste and prolonged latency as shown in Figure 7(a). The *prefetcher* is sent too early so that the timing to use the returned grant is already passed when the data packet arrives. The resource is wasted, while the data packet misses the opportunity to reduce its scheduling latency.

Similarly, a late *prefetcher* could also miss the opportunity to reduce the scheduling latency for the data packet, as shown in Figure 7(b). If the *prefetcher* is sent too late after a potential SR that could reduce latency, the data packet might have to wait for scheduling latency as if no *prefetcher* is issued. In the worst case for both early and late *prefetcher*, it may result in missed latency savings up to $T_{sr\_period} + T_{sr\_grant}$.

LDRP's approach LDRP aims at reducing the scheduling latency at marginal radio resource cost. Let a *prefetcher* be sent t_p before the data packet. The parameter t_p must meet two requirements. First, we should ensure $t_p \geq T_{sr\_grant}$. Note that, an SR can only request a grant to be used at T_{sr_grant} after the SR. Therefore, $t_p \geq T_{sr\_grant}$ guarantees that the SR is sent only if it helps to reduce the scheduling latency for the data packet. Second, we must ensure $t_p \leq T_{sr\_grant}$. This is to let the requested grant be used to transmit the data packet. No resource waste or premature SR is incurred.

Consequently, our timing design is to set the time advance as $t_p = T_{sr\_grant}$, which meets both requirements. Note that T_{sr_grant} is typically constant for a BS, being the accumulative latency of SR processing latency + 4ms, where 4ms is a standardized parameter in [28]. In our experiment, more than 96.5% of T_{sr_grant} is identical under a BS regardless of the carrier. If T_{sr_grant} changes after handover to a new BS, we update T_{sr_grant} immediately. If the SR requested by the *prefetcher* is lost, the solution is still no worse than the current practice. Similar to *rouser*, the timing to insert a *prefetcher*, T_{sr_grant}, could be inferred from application layer with techniques in §4.3. Therefore, no root access is necessary.

Impact of the data packet size A *prefetcher* helps reduce scheduling latency if the data packet size ≤ grant - *prefetcher* size, which is common in reality as >99% of initial grants

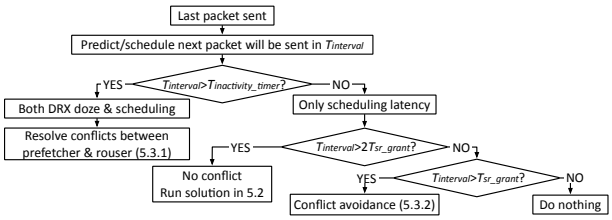


Fig. 9: The workflow of conflict handling in LDRP.

in our experiments exceed 100B in all operators, while the uplink sensory data is smaller than half of that. Therefore, a *prefetcher* initiates an SR, and gets a returned grant that suffices for the data packet to be sent with the *prefetcher*.

However, a corner case arises when the grant in response to SR is enough for the data packet, but not for a *prefetcher* + the data packet. As shown in Figure 8(b), the device could only send the *prefetcher* and a portion of the data packet. A BSR further requests a grant for the remaining data. The data packet thus suffers extra BSR latency compared to the case without *prefetcher* (Figure 8(a)). In the worst-case scenario, this latency increases by T_{sr_grant} (∼8ms). We discuss the probability of this case in Appendix B. However, even in this corner case, the worst case happens only when the data and the *prefetcher* arrive in the same SR period, with probability T_{sr_grant}/T_{sr_wait}. For other conditions in the corner case, the latency is the same as vanilla LTE.

5.3 Handling the Conflicts for Low Latency

LDRP further resolves several conflicts for overall latency reduction. Figure 9 illustrates the workflow of LDRP. Let $T_{interval}$ be the time interval between the last and the next expected packet. LDRP thus reduces various latency elements. It handles improper interplay between latency elements, and between dummy and data packets.

5.3.1 Conflict Resolution Between Latency Elements

LDRP issues two types of dummy packets for latency reduction: *rouser*s for DRX-induced doze latency, and *prefetcher*s for scheduling latency. Figure 10(a) illustrates their conflicts. A *rouser* itself is a dummy message that needs to be sent before a *prefetcher*. Once turning the device to DRX ON, it asks for the grant, which could carry both *rouser* and *prefetcher*. Therefore, the *prefetcher* is sent by the grant requested by the *rouser*. The grant-induced scheduling latency is not reduced at all. The latency penalty can be as large as $T_{sr\_period} + T_{sr\_grant}$ compared to no-conflict case in Figure 10(b).

LDRP's approach and benefits To resolve this conflict, we refine the timing control to ensure both dummy packets' effectiveness. Specifically, we should make sure a *rouser* is sent when a *prefetcher* hits the device buffer, so that the *prefetcher* can take effect and reduce the scheduling latency. A *rouser* takes at most $T_{sr\_period} + T_{sr\_grant}$ to be sent out as a dummy message and a *prefetcher* needs to be sent T_{sr_grant} before the data packet. Therefore, we adapt the timer from $t_p = T_{drx\_doze\_max}$ to $T_{drx\_doze\_max} + T_{sr\_period} + 2T_{sr\_grant}$ to ensure a *rouser* is sent before a *prefetcher*. The *rouser* thus endures $T_{drx\_doze\_max}$ that guarantees the doze is completed and then sent out. The latency penalty is thus reduced to zero.

5.3.2 Conflict Avoidance Between Dummy and Data

The next conflict arises between LDRP's dummy packets and the last legitimate data packet. If a *rouser* conflicts with the last packet, this does not pose an issue: the *rouser* can still help the device to remain in the ON period for $T_{inactivity}$. We thus only discuss where a *prefetcher* intervenes with the last packet. We show how LDRP adapts this for latency reduction.

There are two instances when a *prefetcher* arrives in the buffer *before* the last data packet being completely sent out, shown in Figure 11. In case (a), the *prefetcher* does not provide any latency reduction. The grant for the last packet has enough room to carry the *prefetcher*, which will be sent together. There is no *prefetcher*-requesting grant for the next data packet. In case (b), a *prefetcher* may increase the latency. The grant for the last data packet cannot accommodate the piggy-backed transmission of the *prefetcher*. A BSR request is thus triggered by the device to request for more grants. Since BSR specifies the size for the dummy message *prefetcher*, the returned grant does not suffice to transmit the data packet. This subsequently invokes another round of BSR-grant operations. The data packet might suffer from extra BSR latency.

LDRP's approach and benefits To avoid the conflicts, LDRP adjusts the timing of a *prefetcher*. It leaves enough time for the last packet to complete its transmission before the *prefetcher*. Recall that the theoretical maximum uplink latency that the last packet would experience after optimization is T_{sr_grant}. The dummy *prefetcher* is then sent at least T_{sr_grant} after the application sent its last packet. Specifically, if the time gap (between the last packet arrival and the next packet arrival) is larger than $2T_{sr\_grant}$, we send a *prefetcher* T_{sr_grant} before the next packet. This is the timing we designed in §5.2; it will not break the above condition. Otherwise, we send a *prefetcher* T_{sr_grant} after the last packet. This choice will reduce less latency compared to the timing in §5.2 without conflicts. However, we avoid the cases of Figure 11, where a conflict negatively affects the latency. We thus reduce the maximum optimization lost from T_{sr_grant} to 0.

5.4 Rootless Handover Prediction

Recall that, we cannot mitigate the handover disruption time, as it is rooted in the 4G hard handover design in PHY layer. One solution is to incorporate a handover predictor, which allows users to be notified of such a long disruption time in advance. The notification prepares the users for the handover using techniques such as pre-rendering.

Like other components, LDRP aims to predict handover on the device side without root. Existing works on handover prediction cannot satisfy the requirements. The conventional handover prediction is deployed on the base station side, which needs users' history and global cell coverage information [34], [35], [36], [37], [38]. Several recent works propose solutions to predict handover on the device side, but they require low-level information that cannot be easily acquired without root [32], [39].

We discover that, the LTE handover decision logic is *interactive, stateful and stable*. Appendix D details how the

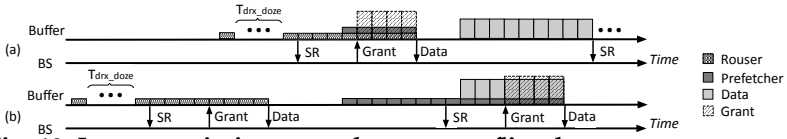


Fig. 10: Improper timing control causes conflicts between components.

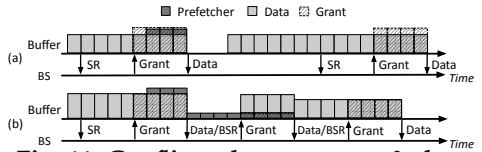


Fig. 11: Conflicts: dummy msgs & data.

Algorithm 1: Handover prediction using measurement reports in LDRP.

Input: Serving cell s and measurement report m which reports an event $m.event$ on cell s_i.
Output: Prediction of handover.

1 **if** *s and s_i have the same frequency and $m.event = A_3$* **then** return **true**;
2 **else if** s and s_i have different frequency and signal strength of s satisfies A_2 and $m.event = A_5$ **then** return **true**;
3 return **false** otherwise;

base station configures the device with the standardized criteria [40] of measuring the current and nearby base stations' radio signal strength. The device measures and reports to the base station if any criteria is satisfied. The base station then determines whether a handover is necessary based on a pre-determined algorithm. While different base stations may have varying configurations for the thresholds of triggering 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.

LDRP leverages it and devises a lightweight predictor based on our experimental data and popular base station's internal decision logic [41], shown in Algorithm 1. It classifies the handover into two categories: *Intra-frequency* handover using the same frequency band, and *inter-frequency* handover using different 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). Otherwise, Inter-frequency handover is triggered when the current base station's signal strength is weak (event A2), and a nearby base station's signal strength is satisfactory (event A5). LDRP monitors these events to predict the handover, which offers high accuracy in reality. 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 analysis. LDRP prediction algorithm is accurate as it approximates the base stations' handover decisions. In addition, it is robust to noisy wireless channels. Even if a handover is missed or a normal event triggers a false alarm, LDRP is no worse than legacy LTE.

∘ *Robustness to dynamic/noisy wireless channels.* Our prediction is robust to channel dynamics and noises. Algorithm 1 is based on measurement-triggered *events*, rather than direct radio measurements. To handle transient dynamics, the standards [40] have defined thresholds for event triggering (in Appendix D). False measurements due to radio fluctuations/noises are thus mitigated, resulting in a robust prediction.

∘ *Tackling false positives/negatives.* If handover is mispredicted, LDRP ensures that VR experiences latency no larger than 4G LTE. There are two scenarios. First, LDRP may predict a handover that would not happen. Both network and app-level adaptations do not incur extra latency or overhead. Second, LDRP may miss a handover. This rolls back to the standard LTE and no extra latency is occurred.

Predict measurement reports from at the APP layer Measurement reports provide a strong indication to predict HO. If LDRP detects that the device has root access, it acquires this low-level information from modem with MobileInsight to accurately predict handover. To serve the users without such privilege, LDRP further infers measurement reports using OS APIs, which demand no root permission or firmware access (e.g., Android API).

To achieve so, we make an observation that measurement report configurations in a single cell remain variant for all users. Besides, they are infrequently changed over time. This is because that, the handover config for a cell is mostly determined by local cell deployment. We construct a database that records $< cell\_id, measurement\_report\_configuration >$ tuples. For a device, LDRP checks whether the root access is available. If so, it directly acquires such info from debug port and reports a new data point to the database. The database will replace the cell configuration with the latest copy. Otherwise, LDRP gets the id for the current serving cell from OS and queries its configuration from the database. With the parameters, LDRP uses OS API to learn the real-time signal strength and infer the measurement report for HO prediction.

Prediction-enabled latency masking. The handover prediction helps mask latency at app levels. It notifies apps for early adaption (e.g., pre-rendering or FPS adaption [8], [42]). The concrete masking technique is out of the scope of this paper. The notification is sent to apps via OS APIs.

6 LDRP Discussion and 5G Applicability

We analyze LDRP and extend the discussion to irregular traffic and 5G.

6.1 Miscellaneous Issues

Energy Analysis LDRP incurs extra energy overhead from four sources. First, transmitting a *rouser* incurs a longer ON period. The time to send a *rouser* can be as long as $T_{sr\_period} + T_{sr\_grant}$. It incurs $T_{sr\_period}/2 + T_{sr\_grant}$ on average. Second, T_{drx_doze} is not predictable so we select $T_{drx\_doze\_max}$ to prioritize latency over energy. The extra ON period is $\delta = T_{drx\_doze\_max} - T_{drx\_doze}$ for each *rouser*. If the packet arrives during DRX OFF, $T_{drx\_doze\_max}$ equals to T_{drx_doze} and $\delta = 0$. Otherwise, $T_{drx\_doze} = 0$ and $\delta = T_{drx\_doze\_max}$. The expectation of δ is thus

$p_{on} \cdot T_{drx\_doze\_max}$, where p_{on} is the probability of a packet arriving during DRX ON period. If we ignore background traffic and assume the packet arrives in the buffer at a random time, p_{on} = onDurationTimer / DRX cycle. Third, an early *rouser* (due to inaccurate estimation) also causes a longer ON period. Denote ϵ as the estimation error. When the *rouser* arrives during DRX OFF, the extra ON period is ϵ. Otherwise, ϵ incurs no extra ON period. Finally, sending extra small messages incurs extra energy waste.

Impact on the network spectrum efficiency For every data packet, we define its spectrum efficiency $SE = \frac{\text{sizeof (data packet)}}{\text{sizeof (Total UL resource granted)}}$. When a data packet suffers from doze latency and LDRP sends a *rouser*, it reduces SE by half: the *rouser* and the *prefetcher* initiate two grants, while the legacy LTE only requests for one. The extra grant occupies ≈2 RB in commercial networks. LDRP trades-off SE for low latency. When LDRP sends a *prefetcher* only, two scenarios arise. In the normal case, the grant from SR can carry both the data packet and a *prefetcher*. Therefore, LDRP requests no extra grant and SE is the same as the legacy LTE. In the corner case discussed in §5.2, the BS allocates at most sizeof (*prefetcher*) extra grant. One extra RB is thus wasted, since a single RB is sufficient to carry a *prefetcher*. SE is reduced by $\frac{\text{sizeof (}prefetcher\text{)}}{\text{sizeof (}prefetcher\text{ + grant from SR)}}$. This value multiplying the probability of the corner case (see Appendix B) yields the expectation of SE reduction.

Impact of background traffic LDRP still reduces latency in the presence of background traffic. No matter whether the background packet is sent before a *rouser* or between a *rouser* and a data packet, the *rouser* will keep the data packet at the DRX ON state, thus eliminating the DRX doze latency. On scheduling latency, if the background traffic is sent after the data packet, it does not affect the *prefetcher*. If the background traffic is in between, the *prefetcher* reduces its latency, which indirectly reduces latency for the real data packet. It still does not increase the latency compared to legacy LTE without LDRP. When the background traffic is sent before a *prefetcher*, it will be sent out through BSR before the data packet in the worst case, equivalent to no optimization.

What if the uplink traffic is not strictly regular? While mobile sensors produce regular data packets, the actual uplink data packets might not be strictly periodic. This can be caused by mobile OS overhead, prediction inaccuracy, or sensor periodicity variance. LDRP still guarantees no worse latency than legacy LTE, and saves LTE latency in most scenarios. We show the following Theorem 6.1 and prove it in Appendix C.

Theorem 6.1. *For the data packet that should have arrived at $T_{interval}$ but actually arrives at T, LDRP does not incur extra latency compared with the legacy 4G LTE.*

LDRP for multiple applications LDRP can reduce latency for multiple applications on a single device that use its APIs. LDRP treats them equally as if from the same application. For each request, LDRP checks the interval between the packet and the last packet and makes the decision of which component to use, as shown in Figure 9. This will not incur much additional overhead, as LDRP will ensure the dummy packets are sent only when necessary without conflict.

Impact on other applications in the device that do not use LDRP An application which does not use LDRP will not be negatively impacted by an application that does so in the same device. Instead, it might even benefit from LDRP despite not calling its API. For simplicity, we denote a packet Y that uses LDRP and a packet X that does not. 1) If X is sent after Y. In this case, Y uses LDRP to finish transmission earlier. This will not affect X's transmission. Instead, it might accelerate it as the scheduled request for Y can now be used to deliver X. 2) If X is sent before Y. In this case, Y might incur dummy packets using LDRP. If the small packets are later than X, the X's transmission is not affected. Otherwise, the dummy packets can help reduce X's doze or scheduling latency, benefiting its performance.

LDRP for non-regular traffic For ML/AI apps in §2.2 with irregular but predictable uplink traffic, LDRP works equally well. we do not recommend LDRP for apps with irregular yet unpredictable uplink traffic. If users intend to use our APIs, latency reduction cannot be ensured.

Network side overhead LDRP incurs little overhead on the network side. The overhead stems from processing extra signaling, which is marginal compared with normal operations. This is because BS monitors the control and data channels continuously, regardless of whether it receives SR.

Impact on other users in the same cell We first note that LDRP does not modify the scheduling policy or operations. Instead, the device asks for grant using data packets. This is standard-compliant as the dummy packets will still be sent and charged. Besides, if the devices under a BS all use LDRP, they will still benefit from LDRP. The core idea of LDRP is to schedule a device's allocated resources in advance if its data arrival can be predicted. The procedure does not sacrifice other users' access in general. Moreover, if a certain device does not adopt LDRP, its latency may be slightly prolonged. This arises when the BS assigns the last available resource to an LDRP user who advances its scheduling, while this resource could have been available to the non-LDRP user. However, the impact is minimal, as the BS will serve the user the next slot (1ms) and the throughput is not affected.

6.2 LDRP is applicable to 5G

In principle, LDRP is applicable to 5G, which has three usage cases. Enhanced Mobile Broadband (eMBB) extends 4G technology. Massive Machine Type Communication (mMTC) is for cellular IoT, whose design is based on LTE-M and NB-IoT [43]. The scheduling mechanisms and handover logic of both modes largely remain unchanged [40], [44]. Consequently, LDRP can still diagnose latency components and reduce latency for the 5G scenarios. Ultra Reliable Low Latency Communications (URLLC) targets low-latency communication. Although not fully standardized, the potential grant-free scheduling might partially achieve LDRP's latency reduction for scheduling latency. However, LDRP's latency diagnosis, DRX doze latency reduction, and handover prediction will still help URLLC applications.

7 Implementation

We implement LDRP as a standalone user-space daemon with Android NDK. A similar implementation is also feasi-

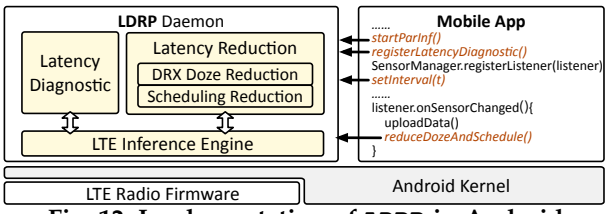


Fig. 12: Implementation of LDRP in Android.

ble for iOS. Figure 12 shows its key components, including an inference engine that offers key LTE parameter inference, a latency manager for latency diagnosis and reduction with conflict resolution, and a set of APIs for latency-sensitive applications. To use LDRP, a latency-sensitive mobile app requests LDRP service using its APIs as detailed below. At runtime, LDRP first detects if the device connects to a new base station by checking the change of serving cell ID. Upon cell changes, LDRP starts to infer the key LTE parameters for this new cell. Once they are obtained, LDRP initiates its latency manager to diagnose and reduce latency.

APIs LDRP provides easy-to-use application-layer APIs for mobile application developers. Figure 12 showcases these APIs with a mobile VR application. The app first calls startParInf() so that LDRP daemon starts and infers the LTE parameters relevant to latency reduction components. The daemon detects possible parameter changes (say, upon handover) and re-runs the inference procedure whenever necessary. The app can then call registerLatencyDiagnostic() to start the diagnostics functionality. As our VR application uploads periodic sensory data packets, it calls setInterval(t) to inform LDRP such periodicity. Whenever a data packet is sent, the application calls reduceDozeAndSchedule() for LDRP to reduce latency for the next packet.

LTE inference engine It first infers the key LTE parameters for LDRP's latency reduction based on the approaches in §4.3. We use DNS requests/responses as probing packets, which have low deployment cost (by using LTE's readily-available DNS servers) and higher accuracy (compared to other probing packets delivered with low priority such as ICMP). For DNS servers, LTE assigns its own in-network DNS server when the device attaches to it, which provides fast and stable service. We use such DNS servers for our experiment.

In addition, the inference engine predicts HO as described in §5.4. It interacts with the centralized server to query for the latest measurement report configurations for the current cell stored in the database. When a HO happens, it re-query the results. It then actively leverages the *SignalStrength* class to get the signal strength for the current and neighboring cells. When the measurement results satisfy the condition, it predicts a measurement report. When the reports meet the condition as shown in Algorithm 1, it predicts a handover and sends a broadcast Android notification.

Moreover, we note that simply running the inference in §4.3 may be inaccurate in practice, since it is sensitive to the noises from background traffic, vendor-specific base station behaviors, and server load. To this end, we optimize our implementation to mitigate these noises and improve inference accuracy. Specifically, we add a few filters to reduce the noise. For instance, when measuring the scheduling-related parameters, we know that T_{rtt} should be greater than 4ms in reality, therefore, if the packet response pair is received within 4ms, we ignore this round of experiment.

Latency manager It realizes the latency diagnostics in §4 and reduction in §5. A practical issue to realize them is to optimize the dummy packet's construction and delivery for low cost. Both *prefetcher* and *rouser* messages in LDRP should be as small as possible so that extra data overhead is minimized. In addition, a smaller *prefetcher* will decrease the likelihood of the corner case discussed in §5.2. The smallest packet we could generate in the Android device without root is an ICMP ping packet with IP header only via system command. Our implementation issues only one small ICMP packet to the local gateway in LTE that serves the users.

8 Evaluation

We assess how LDRP diagnoses each latency component and improves the overall latency and QoEs for emergent mobile applications, evaluate the effectiveness of solution components in LDRP, and quantify LDRP's overhead.

Experimental setup We run LDRP on Google Pixel, Pixel 2, Pixel XL, and Pixel 5. We quantify the latency diagnosis and reduction over AT&T, Verizon, T-Mobile, and Sprint. The evaluation covers 355 unique cells. We repeat the tests in static, walking (∼1m/s), and driving (∼30mph) scenarios. We do experiments mostly in metropolitan areas while driving tests cover rural areas as well. The radio signal strength varies from -120 to -80dBm, covering good (>-90dBm), fair ([-105, -90dBm]), and bad (<-105dBm) conditions. To quantify LDRP, we use MobileInsight [26] to extract the ground truth of fine-grained per-packet latency breakdown from the chipset.

To gauge LDRP's impact on the network side, we build a USRP-based testbed. A server with Intel i7-9700k CPU and 32G RAM runs srsRAN [45] for the functions of core network and BS processing. A USRP B210 connects to the server and provides wireless access for the devices. We plug sysmoUSIM [46] into the test phones, and register them.

8.1 Overall Latency Diagnosis and Reduction

We showcase LDRP's latency diagnosis, reduction, and QoE improvements with four representative emergent mobile applications:

∘ *Mobile VR.* We use the showcase VR game as described in §2.2. We measure the latency of the sensor data and control data, and use it to gauge how our design reacts to VR games.

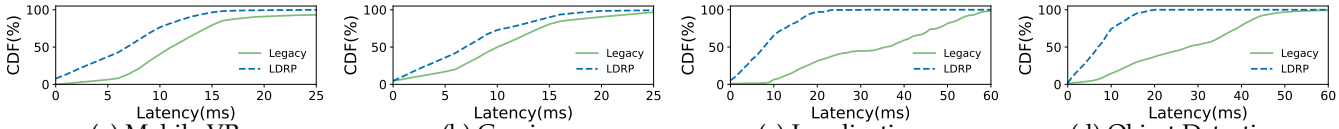
∘ *Localization.* We write an Android app that uploads the periodic GPS location status to the cloud via the Android API [21]. We encode each location update in 22 bytes and send it to the cloud every second.

∘ *Object recognition.* We prototype an object recognition app using MobileNetV2 [47], a phone-based deep learning model. The app processes camera frames and uploads the recognition result to cloud. Inference typically takes 250ms.

∘ *Gaming.* We evaluate its latency by replaying the traces from PUBG Mobile [2], one of the most popular multi-player online mobile games. Since PUBG traffic is not strictly regular, we use it to demonstrate the effectiveness of LDRP

App		AT&T			Verizon			T-Mobile			Sprint		
		Leg	LDRP	η	Leg.	LDRP	η	Leg.	LDRP	η	Leg.	LDRP	η
Mobile VR	Med.	N/A	N/A	N/A	12.0	8.0	0.5×	11.0	6.0	0.8×	N/A	N/A	N/A
	95%	N/A	N/A	N/A	28.0	15.0	0.9×	40.0	14.0	1.9×	N/A	N/A	N/A
Gaming	Med.	10.0	6.0	0.7×	9.0	7.0	0.3×	9.0	7.0	0.3×	17.0	11.0	0.5×
	95%	17.0	15.0	0.1×	15.0	15.0	0×	15.0	15.0	0×	27.0	21.0	0.3×
Localization	Med.	38.0	5.0	6.6×	50.0	14.0	2.6×	42.0	5.0	7.4×	30.5	14.0	1.2×
	95%	46.0	14.0	2.3×	59.0	23.0	1.6×	48.0	10.0	3.8×	61.7	25.8	1.4×
Object Detection	Med.	23.0	7.0	2.3×	38.0	9.0	3.2×	33.0	5.0	5.6×	30.0	15.0	1.0×
	95%	47.8	16.0	2.0×	51.0	15.3	2.3×	45.0	10.0	3.5×	59.0	27.5	1.1×

TABLE 4: Uplink network latency (ms) reduction by LDRP in evaluations with four apps. η=(Legacy-LDRP)/LDRP.



(a) Mobile VR (b) Gaming (c) Localization (d) Object Detection

Fig. 13: LTE latency with and without LDRP in representative apps.

Error rate	AT&T	Verizon	T-Mobile	Sprint
$T_{scheduling}$	3.2%	1.5%	2.0%	3.0%
T_{drx_doze}	3.0%	1.3%	3.0%	1.3%
T_{HO}	4.0%	3.2%	3.4%	2.8%

TABLE 5: Evaluation of LDRP latency diagnosis.

	AT&T	Verizon	T-Mobile	Sprint
Predict HO (Precision)	58%	55%	54%	60%
Predict HO (Recall)	83%	70%	75%	65%

TABLE 6: Evaluation of LDRP handover prediction.

as discussed in §6.1. We use the traffic emulator to send data packets based on the trace.

Accuracy of latency diagnosis We check how accurate LDRP can diagnose each latency element from the application layer. For each cell, we first collect ground truth by analyzing the physical/link messages from MobileInsight. We then use LDRP component to infer the parameters, diagnose the latency, and compare them with the ground truth. We calculate the average error rate for each component. The results are shown in Table 5. As we can see, the inference error rate is at most 4.0% for any parameters in all 4 operators. LDRP diagnosis is accurate as argued in §4.3.

Overall LTE latency reduction Table 4 and Figure 13 show LDRP's latency reduction for these apps in static settings with fair-good signal strength; other scenarios have similar results as detailed in §8.2. On average, LDRP achieves 4-5ms (0.5-0.8×) latency reduction in mobile VR, 8-37ms (1.2-7.4×) reduction in localization, 8-29ms (1.0-5.6×) reduction in object detection, and 1-6ms (0.3-0.7×) reduction in gaming for all 4 LTE carriers. Our breakdown analysis further shows these apps suffer from different latency bottlenecks. For the localization and object detection, the majority of data packets suffer from both DRX doze and scheduling latency. For the VR and gaming with more frequent packets, the scheduling latency is the major latency bottleneck. LDRP can reduce both bottleneck latencies and thus benefit all these applications.

QoE improvement To showcase the impact of LDRP on the mobile VR, we conduct a user study with 10 participants to evaluate the subjective experiences of using VR with/without LDRP. Figure 14 shows the average Mean Opinion Score (MOS) on three aspects: graphical visual quality, responsiveness, and overall experience. Participants rate 1 (Bad) to 5 (Excellent) on these three aspects of the VR game with constant head position changes. The results show that LDRP can improve the visual quality by 8% (3.1→4.0), responsiveness by 63% (2.4→3.9), and overall experience by 46% (2.4→3.5).

5G latency diagnosis and reduction LDRP works for 5G in theory as we discussed in §6.2. We evaluate that LDRP can reduce 5G eMBB latency under AT&T and T-Mobile 5G networks. Since we do not have access to its fine-grained data-plane traces, we measure RTT at the application layer for AT&T (T-Mobile). LDRP reduces RTT by 4.6ms (2.5ms) for Gaming, 20.5ms (32.0ms) for Localization, and 19.8ms (27.0ms) for Object Detection. The results are similar to the latency reduction in their 4G networks. Although we cannot directly measure the diagnosis accuracy without viable tool to capture and decode 5G logs, the reduction experiment indirectly verifies that our analysis is accurate, since the reduction is based on the parameters inferred for diagnosis.

8.2 Micro-Benchmarks

We next assess LDRP's solution components under various signal strengths and user mobility patterns.

DRX-induced latency reduction (§5.1) As shown in §5.1, LDRP helps reduce the DRX doze latency if the inter-packet interval larger than $T_{inactivity}$ (otherwise the DRX doze latency is always 0 with/without LDRP). Figure 15 shows LDRP's DRX latency reduction under various signal strengths and mobility patterns. We run this test under the most popular setting of $T_{inactivity}$ = 200ms (Table 3) when the inter-packet interval is $1.5 \cdot T_{inactivity}$. We also test other intervals and get similar results. In all scenarios, LDRP reduces the DRX doze latency to 0 for all LTE carriers. This results in 21–41ms mean latency reduction and 40–57ms 95% latency reduction.

Scheduling latency reduction (§5.2) We next quantify the reduction in uplink scheduling latency. The latency reduction ratio, η, is defined as that of the reduced latency and the LDRP latency. Table 7 shows the results in different carriers, signal strengths, and mobility patterns. In all these

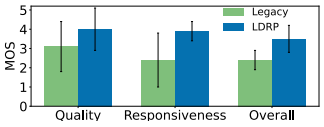


Fig. 14: MOS of mobile VR.

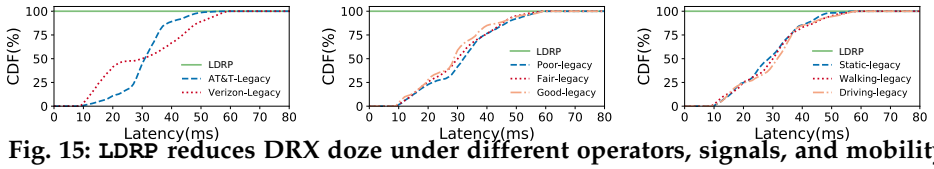


Fig. 15: LDRP reduces DRX doze under different operators, signals, and mobility.

Scenario		AT&T			Verizon			T-Mobile			Sprint		
		Leg.	LDRP	η	Leg.	LDRP	η	Leg.	LDRP	η	Leg.	LDRP	η
Static-Poor	Med.	12.0	5.0	1.4×	12.0	9.0	0.3×	11.0	7.0	0.6×	20.0	12.0	0.7×
	95%	17.0	11.0	0.5×	17.0	17.0	0×	17.0	16.0	0.1×	30.0	26.0	0.2×
Static-Fair	Med.	13.0	8.0	0.6×	11.0	8.0	0.4×	17.0	13.0	0.3×	18.0	14.0	0.3×
	95%	17.0	15.0	0.1×	17.0	13.0	0.3×	12.0	6.0	1.0×	32.0	29.0	0.1×
Static-Good	Med.	13.0	6.0	1.2×	10.0	5.0	1.0×	8.0	6.0	0.3×	13.0	7.0	0.9×
	95%	17.0	11.0	0.5×	17.0	16.0	0.1×	16.0	11.0	0.5×	27.0	18.0	0.5×
Walking	Med.	11.0	8.0	0.4×	13.0	6.0	1.2×	12.0	7.0	0.7×	19.0	13.0	0.5×
	95%	17.0	11.0	0.5×	16.0	16.0	0×	17.0	16.0	0.1×	30.0	26.0	0.2×
Driving	Med.	14.0	8.0	0.8×	14.0	8.0	0.8×	12.0	8.0	0.5×	17.0	13.0	0.3×
	95%	18.0	17.0	0.1×	17.0	11.0	0.5×	17.0	16.0	0.1×	29.0	27.0	0.1×

TABLE 7: Scheduling latency (ms) in four mobile carriers. η=(Legacy-LDRP)/LDRP.

scenarios, LDRP reduces the median scheduling latency by 0.3-2.5×, and reduces the 95th latency by up to 1.7×.

Conflict handling for latency reduction (§5.3) We confirm the effectiveness of LDRP's conflict resolution/avoidance. We adapt LDRP's APIs to enable/disable the conflict handling in §5.3. Table 8 compares the overall latency with/without LDRP's conflict handling. We first illustrate LDRP can resolve *rouser* and *prefetcher* conflict. We use Localization as its traffic pattern satisfies the condition (long interval) for potential conflict. Compared with no conflict resolution, LDRP reduces extra 8.82-60.0% latency in all operators. We next evaluate how LDRP handles data and dummy packets conflicts. The heavy traffic in the Gaming application potentially causes such conflict. We run the Gaming application with LDRP and the APIs without conflict avoidance. Compared with no conflict avoidance, LDRP reduces up to 20% extra latency.

Predicting HO occurrence (§5.4) We check how accurate our rootless HO prediction is. We first collect ground truth by analyzing the RRC-layer signaling messages from MobileInsight. Our implementation then makes prediction when we drive around. Among all operators, LDRP predicts 797 handovers in total, and achieves 54-60% precision and 65-83% recall for each operator. Although the performance is not as accurate as alternative device-side prediction [32], whose recall and precision are both >80%, LDRP achieves the prediction without root privilege or network-side assistance. We have observed two factors that affect the prediction accuracy: (a) *Late API results:* The signal strength reading from Android API is delayed compared to real-time measurement; (b) *Difference of event handling:* Some base stations might change their handover decision over time, making our history-based scheme inaccurate. LDRP prediction is early enough. On average, it predicts the handover 35.9ms ahead of its occurrence.

Conflicts		AT&T	Verizon	T-Mobile	Sprint
rouser & *prefetcher*	w/o Res.	28.0	30.0	34.0	10.0
	LDRP	33.0	36.0	37.0	16.0
	Extra Red.	17.9%	20.0%	8.82%	60%
Data & dummy	w/o Res.	3.5	2.0	2.0	5.0
	LDRP	4.0	2.0	2.0	6.0
	Extra Red.	14.3%	0.0%	0.0%	20%

TABLE 8: Latency (ms) reduction with conflict handle.

	AT&T	Verizon	T-Mobile	Sprint
Extra Data (KB/s)	0.20	0.15	0.41	0.23
Extra Sig. Msg	3.8%	3.7%	4.3%	3.3%
Extra DRX ON period	1.7%	4.2%	5.8%	2.1%

TABLE 9: Overhead of LDRP.

8.3 Overhead

Overhead of dummy messages The dummy messages may incur additional data usage and thus billing. Table 9 shows that LDRP incurs no more than 0.6KB data per second under all carriers. The data overhead depends on the frequency of calling LDRP APIs. For heavy traffic applications (VR, Gaming), the extra overhead is 0.33KB/s while the number for the other two apps is 0.05KB/s. The overhead is acceptable in typical data plans and extra data is only incurred when LDRP APIs are called. As explained in §7, LDRP has minimized the use of dummy for latency reduction.

Extra signaling message The dummy messages incur extra signaling between the device and the BS. We measure this overhead as shown in Table 9. LDRP incurs up to 4.3% messages, which are marginal compared with the total volume of signaling messages. Reducing latency for apps with DRX doze generates more messages. LDRP incurs on average 1.6 extra signaling messages per second for Location and Object Detection. While the other two apps with LDRP generate 0.8 extra message every second on average.

Energy consumption While LDRP exploits the DRX for lower latency, it still respects the LTE's energy saving with accurate timing control and incurs marginal energy cost. We first compare the percentage of the extra DRX ON period with and without LDRP. We track the CDRX events with MobileInsight. As shown in Table 9, for all carriers, at most 5.8% of extra ON period is invoked.

Furthermore, we directly measure the extra energy cost.

We fully charge the device and run Object Detection (with DRX doze) and VR (no DRX doze) applications for one hour, and compare the energy consumption with or without LDRP (under Verizon). With LDRP, two applications incur 2.5% (16.12% to 16.52% of total battery) and 1.0% (37.04% to 37.40% of total battery) extra battery consumption, respectively. This overhead is marginal, as we adjust the timing of *rouser*s to reduce unnecessary energy waste. We also run both applications with Object Detection in the backend. The consumption is 1.8% (40.10% to 40.82%)

Network impact We measure the network impact of LDRP in our SDR testbed. Even in the absence of data transfer, the server spends 0.055ms on average to process the collected signal in every subframe (1ms). In contrast, processing LDRP's extra signaling costs 0.002ms, about 3.6% extra overhead.

Impact on other users or applications We first examine whether LDRP affects another application in the same device that does not use LDRP. We run two applications simultaneously: an active VR application and an emulator with VR traffic that runs on the background. The emulator does not use LDRP. The VR application uses LDRP first and then stops using it. We compare the latency experienced by the emulator when VR uses LDRP or not. Its latency is actually reduced from 10.1 to 8.5ms. As we discussed in §6, the dummy packets from one application can even benefit another application that does not use LDRP.

We next examine whether LDRP affects those non-LDRP users in the same cell. We test a two-device scenario, with both running the Gaming app. Device A never uses LDRP, whereas device B turns on/off LDRP in the test. When B does not run LDRP, A's average uplink network latency is 15.79ms, and the 95th percentile is 24.0ms. When B activates LDRP, A's average latency becomes 15.84ms and the 95th percentile is 24.0ms. Both numbers are not visibly affected. Therefore, the latency of non-LDRP device is not affected, regardless of whether the other runs LDRP or not.

9 Related Work

Many cross-layer techniques have been designed to improve user experience and application performance in mobile networks (see [23] for a survey). They use lower-layer information to improve video streaming [24], to optimize Web access [25], [48], [49], [50], [51], to name a few. Most such solutions seek to boost the application-perceived throughput. Other recent proposals detect whether LTE is the bottleneck for applications [52], estimate the radio link speed [53], or examine how LTE configs affect apps [54]. In contrast, we focus on LTE latency-oriented diagnosis and reduction.

Early efforts are also made to diagnose and reduce LTE network latency. They analyze the latency for Web access over LTE [55], [56], devise application-specific solutions to LTE scheduling latency with modified modem firmware [32], measure the impact of DRX upon LTE from the energy perspective [57], and adjust the RRC parameters to reduce data-plane latency with infrastructure update [58]. Recent work [52], [59] also makes device-based throughput prediction for performance improvement. Our work differs from them since we work on the latency elements that cannot be eliminated with higher throughput. Authors from [5], [60] target reducing one-time connection setup latency, while LDRP reduces latency elements for every data packet in the connected state. Other recent efforts seek to refine the 4G/5G network infrastructure [61], [62].

Studies in [32], [34], [35], [36], [37], [38], [39] aim to predict handover for mobility optimization, but they require either access to low level signaling messages or infrastructure-side intelligence. In contrast, we propose an effective solution on the device side application layer.

10 Conclusion

Diagnosing latency bottleneck and reducing it is critical to many delay-sensitive applications, such as mobile AR/VR, mobile gaming, sensing, machine learning, and robot/drone-based image/speech recognition. In mobile networks, reducing uplink latency is more challenging than its downlink counterpart, since it involves multiple latency elements stemming from power-saving, scheduling, on-demand resource allocation, mobility, etc.

We have designed and implemented LDRP, a device-based solution to LTE latency without any infrastructure changes. LDRP does not require root privilege at the device and works with mobile apps directly. It learns the critical LTE parameters to infer the uplink latency components for latency diagnosis. LDRP then leverages them to reduce the latency. It ensures the network latency is no worse than the legacy 4G LTE, and is applicable to 5G with experimental validation. By design, LDRP uses small dummy messages with proper timing control and conflict handling, in order to eliminate unnecessary latency components from scheduling and power-saving operations. Our experiments have confirmed its effectiveness with a variety of mobile apps.

In the broader context, reducing latency poses a more challenging problem than improving throughput for the networked system community. In the mobile network domain, various tricks have been invented for boosting throughput (e.g., massive MIMO, new modulation, mmWave, etc.). This is not the case for latency. Both its fundamental theory and effective practice are lacking. Moreover, exploring pure device-based solution, which does not require root privilege, offers a nice complement to the infrastructure-centric design, which typically takes years to be deployed.

References

[1] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," *IEEE Transactions on Mobile Computing*, 2019.

[2] PUBG, "PUBG," https://www.pubg.com/, Mar. 2020. [Online]. Available: https://www.pubg.com/

[3] G. Grassi, K. Jamieson, P. Bahl, and G. Pau, "Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–14.

[4] R. Netravali, A. Sivaraman, J. Mickens, and H. Balakrishnan, "Watchtower: Fast, secure mobile page loads using remote dependency resolution," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 430–443.

[5] Y. Li, Z. Yuan, and C. Peng, "A control-plane perspective on reducing data access latency in lte networks," ser. MobiCom '17. New York, NY, USA: ACM, 2017, pp. 56–69. [Online]. Available: http://doi.acm.org/10.1145/3117811.3117838

[6] 3GPP, "TS36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification," Sep. 2019. [Online]. Available: http://www.3gpp.org/DynaReport/36321.htm

[7] O. A. Cloud, "Open AR Cloud," https://www.openarcloud.org/, Mar. 2020. [Online]. Available: https://www.openarcloud.org/

[8] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, 2015, pp. 151–165.

[9] T. Braud, F. H. Bijarbooneh, D. Chatzopoulos, and P. Hui, "Future networking challenges: The case of mobile augmented reality," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1796–1807.

[10] G. Daydream, https://arvr.google.com/daydream/. [Online]. Available: https://arvr.google.com/daydream/

[11] S. Shi, V. Gupta, and R. Jana, "Freedom: Fast recovery enhanced vr delivery over mobile networks," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 130–141.

[12] U. T. Report, https://www.tweaktown.com/news/74682/index.html. [Online]. Available: https://www.tweaktown.com/news/74682/unity-is-about-to-go-public-with-new-ipo-1-2-billion-market-cap/index.html

[13] U. R. Streaming, https://github.com/Unity-Technologies/UnityRenderStreaming. [Online]. Available: https://github.com/Unity-Technologies/UnityRenderStreaming

[14] W. for Unity, https://github.com/Unity-Technologies/com.unity.webrtc. [Online]. Available: https://github.com/Unity-Technologies/com.unity.webrtc

[15] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 155–168.

[16] Amazon, "Amazon Alexa," https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011, Mar. 2020. [Online]. Available: https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011

[17] A. Siri, "Siri," https://www.apple.com/siri/, Mar. 2020. [Online]. Available: https://www.apple.com/siri/

[18] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[19] Waze, "Waze," https://www.waze.com/, Mar. 2020. [Online]. Available: https://www.waze.com/

[20] Uber, "Uber," https://www.uber.com/, Mar. 2020. [Online]. Available: https://www.uber.com/

[21] Google, "Google API for Android," Mar. 2020. [Online]. Available: https://developers.google.com/android

[22] M. Afanasov, A. Djordjevic, F. Lui, and L. Mottola, "Flyzone: A testbed for experimenting with aerial drone applications," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 67–78.

[23] B. Fu, Y. Xiao, H. J. Deng, and H. Zeng, "A survey of cross-layer designs in wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 110–126, 2013.

[24] A. Zhou, H. Zhang, G. Su, L. Wu, R. Ma, Z. Meng, X. Zhang, X. Xie, H. Ma, and X. Chen, "Learning to coordinate video codec with transport protocol for mobile video telephony," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[25] X. Xie, X. Zhang, and S. Zhu, "Accelerating mobile web loading using cellular link information," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, 2017, pp. 427–439.

[26] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang, "Mobileinsight: Extracting and analyzing cellular network information on smartphones," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, 2016, pp. 202–215.

[27] M. Abrash, "What VR Could, Should, and almost certainly Will be within two years," http://media.steampowered.com/apps/abrashblog/Abrash\%20Dev\%20Days\%202014.pdf, 2014.

[28] 3GPP, "TS36.213: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," Sep. 2019. [Online]. Available: https://www.3gpp.org/dynareport/36213.htm

[29] M. Yang and T. Chin, "Scheduling request during connected discontinuous reception off period," Jul. 20 2017, uS Patent App. 14/996,153.

[30] 3GPP, "TS25.331: Radio Resource Control (RRC); Protocol specification," 2006. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25331.htm

[31] ——, "TS36.133: Requirements for support of radio resource management," 2020. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36133.htm

[32] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu, "Supporting mobile vr in lte networks: How close are we?" in *ACM SIGMETRICS*, 2018.

[33] Qualcomm, "QxDM Professional - QUALCOMM eXtensible Diagnostic Monitor," http://www.qualcomm.com/media/documents/tags/qxdm.

[34] H. Ge, X. Wen, W. Zheng, Z. Lu, and B. Wang, "A history-based handover prediction for lte systems," in *2009 International Symposium on Computer Network and Multimedia Technology*. IEEE, 2009, pp. 1–4.

[35] J. F. Abuhasnah and F. K. Muradov, "Direction prediction assisted handover using the multilayer perception neural network to reduce the handover time delays in lte networks," *Procedia computer science*, vol. 120, pp. 719–727, 2017.

[36] M. Ozturk, M. Gogate, O. Onireti, A. Adeel, A. Hussain, and M. A. Imran, "A novel deep learning driven, low-cost mobility prediction approach for 5g cellular networks: The case of the control/data separation architecture (cdsa)," *Neurocomputing*, vol. 358, pp. 479–489, 2019.

[37] X. Chen, F. Mériaux, and S. Valentin, "Predicting a user's next cell with supervised learning based on channel states," in *2013 IEEE 14th workshop on signal processing advances in wireless communications (SPAWC)*. IEEE, 2013, pp. 36–40.

[38] R. A. Paropkari, A. Thantharate, and C. Beard, "Deep-mobility: A deep learning approach for an efficient and reliable 5g handover," in *2022 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*. IEEE, 2022, pp. 244–250.

[39] A. Hassan, A. Narayanan, A. Zhang, W. Ye, R. Zhu, S. Jin, J. Carpenter, Z. M. Mao, F. Qian, and Z.-L. Zhang, "Vivisecting mobility management in 5g cellular networks," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 86–100.

[40] 3GPP, "TS36.331: Radio Resource Control (RRC)," 2020. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36331.htm

[41] "Huawei handover algorithm," 2014, https://www.slideshare.net/herobinh/08102014huawei-handovershandoveralgo.

[42] K. Boos, D. Chu, and E. Cuervo, "Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016, pp. 291–304.

[43] P. Andres-Maldonado, P. Ameigeiras, J. Prados-Garzon, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Narrowband iot data transmission procedures for massive machine-type communications," *IEEE Network*, vol. 31, no. 6, pp. 8–15, 2017.

[44] 3GPP, "TS38.331: NR; Radio Resource Control (RRC); Protocol specification," Oct. 2019. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/38331.htm

[45] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: An open-source platform for LTE evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, 2016, pp. 25–32.

[46] sysmocom, "sysmoUSIM-SJS1 SIM + USIM Card (10-pack)," http://shop.sysmocom.de/products/sysmousim-sjs1, 2016.

[47] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

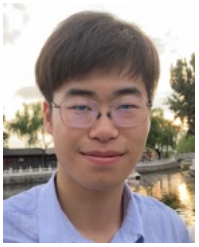
[48] R. Netravali, V. Nathan, J. Mickens, and H. Balakrishnan, "Vesper: Measuring time-to-interactivity for web pages," in *15th USENIX Symposium on Networked Systems Design and Implementation*, 2018, pp. 217–231.

[49] R. Netravali and J. Mickens, "Prophecy: Accelerating mobile page loads using final-state write logs," in *15th USENIX Symposium on Networked Systems Design and Implementation*, 2018, pp. 249–266.
[50] Y. Cao, J. Nejati, A. Balasubramanian, and A. Gandhi, "Econ: Modeling the network to improve application performance," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 365–378.
[51] C. Kelton, J. Ryoo, A. Balasubramanian, and S. R. Das, "Improving user perceived page load times using gaze," in *14th USENIX Symposium on Networked Systems Design and Implementation*, 2017, pp. 545–559.
[52] A. Balasingam, M. Bansal, R. Misra, K. Nagaraj, R. Tandra, S. Katti, and A. Schulman, "Detecting if lte is the bottleneck with bursttracker," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–15.
[53] N. Bui, F. Michelinakis, and J. Widmer, "Fine-grained lte radio link estimation for mobile phones," *Pervasive and Mobile Computing*, vol. 49, pp. 76–91, 2018.
[54] F. Kaup, F. Michelinakis, N. Bui, J. Widmer, K. Wac, and D. Hausheer, "Assessing the implications of cellular network performance on mobile content access," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 168–180, 2016.
[55] B. Pourghassemi, A. Amiri Sani, and A. Chandramowlishwaran, "What-if analysis of page load time in web browsers using causal profiling," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 2, pp. 1–23, 2019.
[56] Z. Yuan, Y. Li, C. Peng, S. Lu, H. Deng, Z. Tan, and T. Raza, "A machine learning based approach to mobile network analysis," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2018, pp. 1–9.
[57] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, 2012, pp. 225–238.
[58] G. Pocovi, I. Thibault, T. Kolding, M. Lauridsen, R. Canolli, N. Edwards, and D. Lister, "On the suitability of lte air interface for reliable low-latency applications," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.
[59] J. Lee, S. Lee, J. Lee, S. D. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee *et al.*, "PERCEIVE: Deep Learning-based Cellular Uplink Prediction Using Real-Time Scheduling Patterns," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys'20)*, 2020, pp. 377–390.
[60] E. Cohen and H. Kaplan, "Prefetching the means for document transfer: A new approach for reducing web latency," in *INFOCOM 2000*, vol. 2. IEEE, 2000, pp. 854–863.
[61] A. Jain, N. Sadagopan, S. K. Lohani, and M. Vutukuru, "A comparison of sdn and nfv for re-designing the lte packet core," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2016, pp. 74–80.
[62] A. Patel, M. Vutukuru, and D. Krishnaswamy, "Mobility-aware vnf placement in the lte epc," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2017, pp. 1–7.

Zhaowei Tan is currently a PhD student in computer science at the University of California, Los Angeles (UCLA), advised by Prof. Songwu Lu. He is broadly interested in systems and security, especially building fast and secure systems that support emerging applications (AR/VR, gaming, AI, etc.) Prior to UCLA, he got his BS degree in CS from Shanghai Jiao Tong University (SJTU). He is a member of the Wireless Networking Group.

Jinghao Zhao received the bachelor's degree from Shanghai Jiao Tong University in 2018. He is currently pursuing the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), Los Angeles. His research interests include wireless systems, mobile computing, and network security. He is a member of the Wireless Networking Group.

Yuanjie Li is currently an assistant professor at the Institute for Network Sciences and Cyberspace at Tsinghua University. He received Ph.D. in Computer Science at University of California, Los Angeles (UCLA) in June 2017 and B.E. in Electronic Engineering at Tsinghua University in 2012. His research interests include networked systems, mobile computing, and network security.



Yifei Xu received his B.S. degree from Peking University in 2021. He is currently pursuing a Ph.D. degree in computer science at University of California, Los Angeles. His research interests include networked systems, mobile computing, and network security. He is a member of the Wireless Networking Group.

Yunqi Guo received the Master degree in computer science from the University of California, Los Angeles (UCLA), CA, USA, in 2018. He is currently working toward the Ph.D. degree at UCLA. His research interests include edge computing, network security, and augmented reality.

Songwu Lu is currently a Professor of computer science with the University of California at Los Angeles. His research interests include mobile networking and systems, cloud computing, and network security.