Bridging the Data Charging Gap in the Cellular Edge

Yuanjie Li, Kyu-Han Kim, Christina Vlachou, Junqing Xie
Hewlett Packard Labs
{yuanjiel,kyu-han.kim,christina.vlachou,jun-qing.xie}@hpe.com

ABSTRACT

The 4G/5G cellular edge promises low-latency experiences anywhere, anytime. However, data charging gaps can arise between the cellular operators and edge application vendors, and cause over-/under-billing. We find that such gap can come from data loss, selfish charging, or both. It can be amplified in the edge, due to its low-latency requirements. We devise TLC, a Trusted, Loss-tolerant Charging scheme for the cellular edge. In its core, TLC enables loss-selfishness cancellation to bridge the gap, and constructs publicly verifiable, cryptographic proof-of-charging for mutual trust. We implement TLC with commodity edge nodes, OpenEPC and small cells. Our experiments in various edge scenarios validate TLC's viability of reducing the gap with marginal latency and other overhead.

CCS CONCEPTS

• **Networks → Mobile networks**; **Network economics**; **Mobile and wireless security**; **Protocol testing and verification**;

KEYWORDS

Intelligent wireless edge, 4G LTE and 5G, charging gap, low latency, loss-selfishness cancellation, proof of charging, network economics

ACM Reference Format:
Yuanjie Li, Kyu-Han Kim, Christina Vlachou, Junqing Xie. 2019. Bridging the Data Charging Gap in the Cellular Edge. In *Proceedings of SIGCOMM '19: 2019 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '19).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3341302.3342074

1 INTRODUCTION

The intelligent wireless edge is redefining our digital lives. By shifting the computation from cloud to edge, it promises ultra low latency for the emergent applications, such as real-time video analytics, safe self-driving, virtual reality, online games, and many more. Since 2017, the wireless edge has been under active standardization [1, 2] and early deployment [3–5].

Many wireless edge scenarios need low-latency service in the indoor/outdoor, and static/mobile settings. A promising solution is the cellular edge. To date, the 4G LTE and emerging 5G cellular networks are the largest wireless infrastructures that offer ubiquitous coverage and seamless mobility support. They operate in the licensed spectrum and support carrier-grade QoS for low latency. We have seen some early operational edge services via 4G/5G (§2.2).

The cellular network data is not free. To enjoy the low-latency services, the edge app vendors should pay the cellular operators by traffic usage (similar to our mobile data plans). Even for the "unlimited" data plans, the edge app's network speed will be throttled (e.g., 128Kbps) if its usage exceeds pre-defined quota. This model inherits from 4G/5G [6, 7], and is standardized in the edge [8].

However, the cellular operator and edge app vendor may not always agree on how much data should be paid. A *data charging gap* can arise, i.e., the data volume charged by the cellular operator differs from what the edge sends/receives. Such gap has been repetitively experienced by mobile users and operators since the 3G era, and has been validated with large-scale measurements [9–14] and even lawsuits [15]. It can be amplified in the delay-sensitive edge, with its adoption of UDP-based real-time protocols and the significantly increased data transfer (e.g. graphical frames in virtual reality and 4K video streaming) in 5G. Our experiments over carrier-grade LTE show that, the charging gap can vary, contributing 8.28MB/hr (8.3%), 59.04MB/hr (6.7%), and 80.64MB/hr (8.0%) in web camera (WebCam) streaming via real-time streaming protocol (RTSP), WebCam streaming via UDP, and virtual reality offloading via GigE vision stream protocol (GVSP), respectively. But with intermittent wireless connectivities or congestion, the gap goes up to 98.16MB/hr, 252MB/hr, and 982.8MB/hr. The gap not only causes over (under)-billing, but also harms the fundamental trust between cellular operators and edge app vendors.

In general, the charging gap can arise from data loss, the selfish charging claims from the operator or edge app vendor, or both (§3.1). The lost data can arise from various network layers, such as the physical-layer intermittent wireless connectivity, link-layer device mobility, IP-layer congestion, transport-layer retransmission, application-layer data drop (e.g. violation of service-level agreements), and more. The selfish charging can be driven by the operator (edge app vendor)'s *economic incentives* for over-(under-)billing. It is hard to eliminate the charging gap for two reasons: (1) For the loss-induced gap, there is a fundamental loss-latency tradeoff (§3.3): Closing such gap will unavoidably delay the traffic (not preferred by edge); (2) The operator (edge) has the economic incentives to over(under)-claim the usage. Such selfish charging is hard to detect: It can be indistinguishable from the data loss. While we believe most operators and edge app vendors are well-behaved, they cannot prove to each other if they are honest, whether the gap is from the data loss or selfishness, and how large each portion is.

We devise TLC, a **T**rusted, **L**oss-tolerant **C**harging scheme for the cellular edge. To bridge the gap, TLC devises the *loss-selfishness cancellation* scheme (intuition in Figure 6 and §4): It lets the cellular operator and edge app vendor negotiate their (possibly selfish) charging claims, during which the data loss can be cancelled out. This scheme has three provable properties. First, different from

SIGCOMM '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-5956-6/19/08…$15.00
https://doi.org/10.1145/3341302.3342074

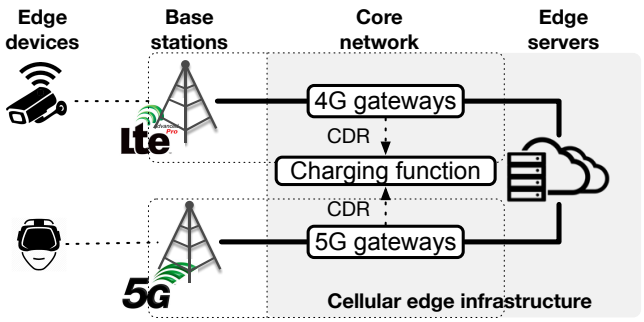


Figure 1: The cellular edge and charging systems.

the legacy 4G/5G, TLC always enforces *bounded* data charging and avoids arbitrary over-/under-billing by the operator/edge. Second, it converges to the correct charging volume when both the edge and operator selfishly minimize/maximize their billing. Third, TLC is latency friendly: It can achieve the correct charging with *1-round* negotiation. Moreover, it initiates at the end of charging cycle, thus not delaying traffic. After the cancellation, TLC constructs a cryptographic Proof-of-Charging (PoC). The PoC is *publicly verifiable* by the independent 3rd-parties (e.g., FCC and courts), without auditing the data transfer. TLC inherently offers the economic deployment incentives for the edge app vendor and cellular operator, and is incrementally deployable by reusing their readily-available mechanisms. TLC's core ideas can also be extended to the generic mobile data charging between the operator and mobile users (§8).

We have prototyped TLC with commodity edge nodes, OpenEPC [16] and Qualcomm small cells. Compared with the legacy 4G/5G, TLC offers provable charging bounds. It reduces the average gap by 80.2% in WebCam streaming, 87.5% in edge-based VR, and 47.06% in the online gaming with marginal latency overhead. It is also scalable: A single HP Z840 workstation can verify 230K PoCs/hour.

2 THE CELLULAR EDGE

This section introduces a cellular edge architecture (§2.1) and its use cases (§2.2).

2.1 A Cellular Edge Architecture

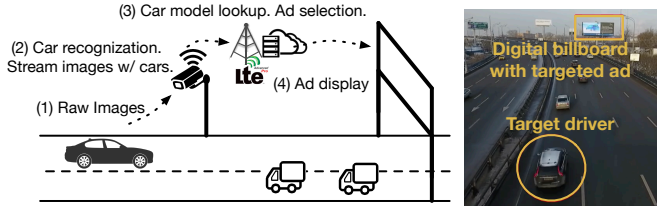
Figure 1 shows the architecture standardized in [1, 6–8, 17], including the edge nodes and the cellular network.

Edge applications: They span on the device and server. The device can be a wireless camera, a vehicle, an IoT gateway, or others. The server can offload the device's computation, run analytics, and aggregate results from devices, etc. The server is deployed in the cellular infrastructure and is owned by the cellular operator or edge app vendor. In both cases, the edge vendor controls its app similar to the cloud.

Cellular-based edge network: The 4G LTE and emerging 5G consist of the radio access network and core network. The radio access network offers wireless access with base stations. The core network bridges the radio access network and the edge servers, monitors the edge's traffic, and charges the edge app vendors. Compared with other wireless (e.g. WiFi and Dedicated Short Range Communications (DSRC)), 4G/5G uses licensed spectrum and scheduling-based primitives, thus facilitating the low-latency edge.

Trace 1 A charging data record (CDR) from 4G LTE gateway.

```
<chargingRecord>
    <servedIMSI>00 01 11 32 54 76 48 F5</servedIMSI>
    <gatewayAddress>192.168.2.11</gatewayAddress>
    <chargingID>0</chargingID>
    <SequenceNumber>1001</SequenceNumber>
    <timeOfFirstUsage>2019-01-07 07:13:46</timeOfFirstUsage>
    <timeOfLastUsage>2019-01-07 08:13:46</timeOfLastUsage>
    <timeUsage>3600</timeUsage>
    <datavolumeUplink>274841</datavolumeUplink>
    <datavolumeDownlink>33604032</datavolumeDownlink>
</chargingRecord>
```



(a) Targeted advertisement: workflow **(b) Deployment [2]**

Figure 2: Real-time, outdoor, targeted ad via LTE edge.

Volume-based cellular data charging: The cellular operators can charge the edge app vendors by their data usage. Similar to the charging to the mobile users, such charging occurs in the core network, and involves the data gateways and charging functions[1]. The data gateway forwards the edge traffic, and generates the charging data record (CDR, exemplified in Trace 1 from OpenEPC in our testbed (§7)). It encodes the device identity (international mobile subscriber identity (IMSI)), the gateway's IP address, the charging policy profile, and the usage. The gateway then sends the CDRs to the charging function. The charging function converts the CDRs to the bills, and may apply policy-driven actions (e.g., high-QoS for low-latency edge traffic, service degrade or network speed limit).

In practice, the operators can apply diverse charging policies based on the data plan and usage. Some may only charge the data that the edge nodes have received, while others may also charge the lost data since it consumes the operators' resource [12]. The operators may charge more for the data with higher QoS priority (§2.2). Some offer the "unlimited" data plan, but throttle the speed if the usage exceeds some quota (e.g. 128Kbps after 15GB [18]). Note that this work does not assume a specific policy. Instead, we focus on the fundamental problem of deciding the usage for any policies.

2.2 Use Cases of the Cellular Edge

We exemplify the use of cellular edge with some real cases.

• **Outdoor targeted advertisement.** Consider the system in Figure 2 (operated in Moscow since 2017 [2]). It offers the digital advertisers *real-time, targeted ads* for highway drivers based on car models. It deploys the roadside cameras ahead of the digital billboard to capture the passing cars' images. Each camera locally runs a car detection and streams the images (with cars) to the edge. Then the edge will classify the car model, and select the appropriate

[1]The data gateways are named as S-GW/P-GW in 4G LTE [6], and UPF in 5G [7]. The charging function is named as CDF in 4G LTE, and CHF in 5G.

advertising to put on the billboard as that car passes. As the car passes, the billboard's ads rotate. Faster targeted ads can rotate more often, allowing billboard advertiser to sell more ads.

We had a conversation with this system's developers, and learned three lessons: *(1) Low latency is crucial:* As the cars move, the system must display the targeted ads *before* a car leaves the billboard. A delayed ad results in the advertisers' revenue loss. The UDP-based protocols (e.g., real-time streaming protocol (RTSP)) may thus be used for the camera streaming. *(2) Wireless via cellular networks:* To simplify the installation and avoid the wired backhaul, this system uses the wireless camera. To support it, 4G/5G is the first choice, with ubiquitous wireless access along the outdoor highway. *(3) Data charging is stressful:* To deliver ads to as many cars as possible, the system operates in 24×7 mode and continuously streams the images. This can result in significant traffic and charging. The advertiser thus wants to save the bill and ensure the operator charges faithfully (no over-bill).

• **Online mobile gaming acceleration:** Tencent cooperates with top-3 Chinese mobile operators to offer the online mobile gaming acceleration in 4G LTE [3]. Many multiplayer online mobile games (e.g., *King of Glory*) need sub-100ms end-to-end latency for smooth player control. The default LTE may not always meet this in congested or outdoor environments. Instead, Tencent's games can request the dedicated, high-QoS session for its player control data (currently only UDP is supported [19])[2]. The game is charged by its request volume. Such scheme is considered as a viable business model and standardized by ETSI [21, 22].

• **Edge-powered augmented/virtual reality (AR/VR):** Envrmnt, a Verizon subsidiary, announced an SDK for low-latency mobile AR/VR [4]. It offloads the mobile headsets' graphical computations to Verizon's 5G wireless edge. The VR/AR edge applications may pay Verizon not only the computation, but also the network data for streaming the graphical frames.

3 CHARGING GAP IN THE CELLULAR EDGE

The *data charging gap* measures the difference of traffic metered by the cellular operator and edge app vendor. We classify the gap causes (§3.1), quantify its impact on edge (§3.2), analyze the challenges (§3.3), and define the problem (§3.4).

3.1 Causes of Charging Gap: A Taxonomy

In general, there are three categories:

Data Loss: The network traffic can be lost during the delivery. The operator may thus charge data that the edge did not receive. Prior large-scale measurements show that, the data loss arises from the cellular network and edge, and spans on multiple layers including (but not limited to): (1) *PHY-layer intermittent connectivity:* When a device temporarily loses its wireless coverage (e.g. signal fluctuates, or device moves to a no-signal zone), the data can be lost over the air [9]; (2) *Link-layer mobility:* The moving device may switch its base stations or radio technologies, in which the data can be lost [10]; (3) *IP-layer congestion:* In congestion, packets can be dropped *after* being charged by the gateway (§3.2); (4) *Transport-layer retransmission:* The data can be over-charged due to spurious retransmission [12]; (5) *Application-layer data drop:* The operator's middle-box may drop the data frames from real-time applications (e.g. video streaming) that exceed the latency requirements or service-level agreements (SLAs) [23, 24]. The loss-induced gap is usually small in general [9, 10]: Most traditional mobile apps use TCP to control the data rate for less loss (e.g. in weak coverage area) and recover the lost data . But this may change in the low-latency edge (§3.2).

[2]LTE defines game-specific QoS Class Indicators (QCI=3/7), which guarantee 50ms/100ms packet delay [20]. The mobile game in the device can invoke Tencent's game APIs and activate such high-QoS session to the game server.

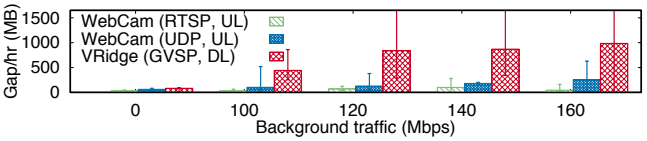


Figure 3: The data charging gap in various congestion levels (RSS ≥ -95dBm, iperf UDP background traffic).

Selfish charging: Both the cellular operator and edge app vendor have economic incentives to cheat the charging system. The cellular operator may over-claim the network usage and over-charge the edge, while the edge app vendor may want to under-claim its usage to save its payment. While unlikely to be common, both are not without foundation. For example, a recent lawsuit from the user claims the operator's over-charging [15], while a recent data bill bargaining [13] is claimed to be user's selfish under-claim. In 4G/5G, the selfish charging volume can be *unbounded*: In the worst case, a dishonest operator could over-charge the edge with arbitrarily high data volume..

Data loss *and* selfish charging: The data charging gap can also be caused by both. For example, if the cellular operator claims 1GB more than the edge, the gap can be due to the loss, the operator's over-claim, the edge's under-claim, or all. It is thus nontrivial to decide the appropriate charging volume.

3.2 How is the Cellular Edge Affected?

The cellular edge apps can be delay-sensitive, thus more vulnerable to the data loss and selfish charging. They may adopt UDP-based real-time protocols (e.g. RTSP for WebCam streaming, proprietary protocols for gaming [19, 25], and GVSP for VR streaming [26, 27]). The data loss may not be always recovered, thus enlarging the gaps. Besides, for the edge apps with 24×7 operations (e.g. targeted ad in §2.2) or heavy traffic (e.g. virtual reality in §2.2), the data usage can be significant. This may amplify the incentives for the operator or edge for selfish charging.

We next quantify both issues. We bear in mind that the experiments of data charging in operational 4G/5G might be detrimental to operators or users. Therefore, we emulate the real edge use cases in §2.2 in our testbed, which operates with carrier-grade LTE core (OpenEPC [16]) and Qualcomm small cells (Figure 11). We test WebCam stream for video analytics using RTSP and legacy UDP, and edge-based VR by replaying the traces from VRidge over operational LTE networks (from [28]). The detailed setup is elaborated in §7.1. To quantify the gap, we record the data usage from LTE gateways and edge device/server every 1s, and compute their gap Δ. Figure 3 and Figure 4 show the charging gaps under various signal strengths and congestions. We make four observations:

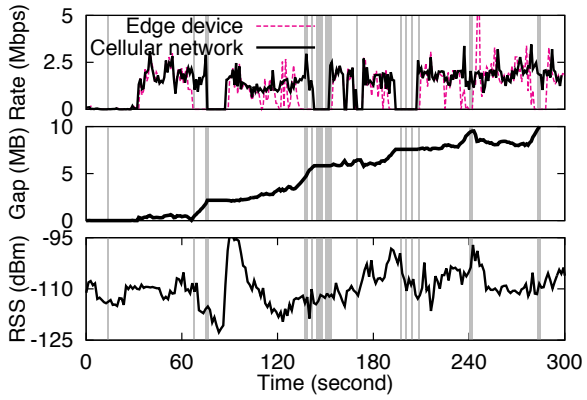


Figure 4: The data charging gap by the intermittent connection (downlink UDP WebCam, no background traffic). The gray areas indicate no uplink and downlink service.

(1) Some edge scenarios involve viable data usage. The average bitrate for RTSP 1920×1080p WebCam stream, UDP-based WebCam and 1920×1080p VR are 0.77Mbps, 1.73Mbps, and 9.0Mbps, respectively. This results in 346.5MB/hr, 778.5MB/hr, and 4.05GB/hr usage. Such volume may offer incentives for the operator and edge app vendor for selfish charging (§2.2);
(2) The loss-induced charging gap is small in ideal scenarios. Figure 3 shows that, in good radio (received signal strength (RSS)≥-95dBm) without congestions, the average gaps are 8.28MB/hr (8.3%), 59.04MB/hr (6.7%), and 80.64MB/hr (8.0%) in the RTSP WebCam streaming, UDP-based streaming, and GVSP edge VR;
(3) Congestion enlarges the gap. Figure 3 shows that, the charging gap tends to increase with the congestion levels. It can go up to 98.16MB/hr, 252MB/hr, and 982.8MB/hr in the RTSP-based WebCam streaming, UDP-based WebCam streaming, and GVSP-based edge VR offloading, respectively.
(4) Intermittent connectivity enlarges the gap. In the experiment in Figure 4, the average wireless dis-connecivity duration (gray areas) is 1.93s. It results in 10.6MB gap in 300s (approximately 127.2MB/hr). The intermittent connectivity can be partially tolerated by buffering the packets (e.g., t=240s, in which the gap decreases). But the buffer is not sufficient to eliminate the gaps. Note that weak signal does not always result in charging gaps: If the device persistently stays the no-signal areas, the network can detect it via radio link failures, detach the device and prevent larger gap. Our LTE core takes 5s on average for this. But it cannot tackle the gaps from the <5s disconnectivity; the gap can still accumulate with repetitive intermittent <5s disconnectivities.

3.3 Why is Bridging the Gap Hard?

To our knowledge, existing solutions cannot handle the data loss and selfish charging *simultaneously*. Partial solutions exist in each class, but are limited by fundamental limitations.

Data loss: Latency-loss tradeoff. Intuitively, the loss-induced gap can be bridged by synchronizing the operator and edge's charging records. This needs a feedback loop, similar to TCP (e.g. proposals in [9, 10, 29]). Unfortunately, such design is unfriendly to the delay-sensitive edge: Before the successful synchronization, the data transfer would be blocked.

We show that, *any* design that seeks to bridge the loss-induced gap by synchronizing the charging records will unavoidably delay the traffic. Bridging the loss-induced gap is fundamentally a distributed counting problem: The edge nodes (cellular gateways) maintain a local traffic counter $\hat{x}_e$ ($\hat{x}_o$), and increase it by 1 for each new packet (or byte). To close the loss-induced charging gap, we should guarantee *consistent* value: $\hat{x}_e = \hat{x}_o$. To decide the charging volume, we should query the edge or network's traffic counter. If the charging gap exists ($\hat{x}_e \neq \hat{x}_o$), the query may be suspended until the gap is closed. Then we have the following impossibility result (proof in Appendix A):

Theorem 1 (Charging Gap V.S. Latency)**.** *Assume the edge app vendor and cellular operator are honest. With arbitrary network data loss, it is impossible for any design to always guarantee (1) consistent view of charging record: $\hat{x}_e = \hat{x}_o$; and (2) every charging volume query will eventually return.*

Theorem 1 is a variant of the CAP theorem [30] from the distributed computing area. The intuition is that, the data loss (which may be charged (§2.1)) is indistinguishable from the no-data-transfer (which should not be charged). Theorem 1 assumes *honest* edge and network. Relaxing this assumption is a double-edged sword: It can further complicate the charging (detailed below), *or* bypass such impossibility (§4).

Selfish charging: No verifiable usage accounting. The selfish charging is hard to avoid for two reasons. First, both the cellular operator and edge app vendor have the *economic incentives* to exploit the data usage. Second, it is technically feasible for them to exploit the charging records. The operator can modify its CDRs for over-billing (validated in our carrier-grade LTE core). The edge can directly modify `netstat` to lower the data volume (root needed), or indirectly reset the bill cycle for smaller usage [31] (no root).

Of course, we believe that most cellular operators and edge app vendors are honest and self-regulated. The challenge is that, neither the operator nor edge can prove to each other or 3rd parties (e.g., FCC) that its charging is honest and trusted[3]. Without verifiable proofs, it is difficult for even the laws to ensure that the network and edge are well-behaved (as shown in the recent lawsuit and debates[13–15]). It is thus difficult for the network and the edge to trust each other.

Data loss *and* selfish charging: Indistinguishability. In general, the gap by the data loss is not always distinguishable from the gap by the selfish charging. For instance, if the cellular operator claims 1GB more downlink traffic than the edge does, the gap can be due to the loss, the operator's over-claim, the edge's under-claim, or all. It is thus nontrivial (if not impossible) to decide whether anyone is cheating, or what is the appropriate charging volume.

3.4 Problem Statement

We aim to bridge the data charging gap in the cellular edge. We assume the cellular operators and edge app vendors can be *honest* (i.e., report the charging correctly), or *selfish but rational* for their own economic interests. In the latter case, the operator aims to maximize charging for the edge, while the edge app vendors seek

[3]The operators may seek to prove its charging with itemized data receipt in the data bill. Unfortunately, this can still be manipulated by a selfish operator with faked data activities; the edge cannot disprove them.

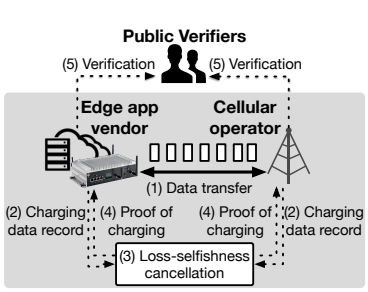


Figure 5: TLC overview.

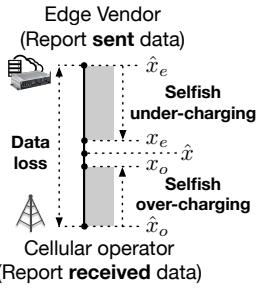


Figure 6: TLC intuition.

Table 1: Notations in TLC

$c \in [0, 1]$	Pre-defined charging weight for the lost data (in the data plan)
T	Charging cycle in data plan: $T = (T_{start}, T_{end})$
$\hat{x}_e$, $\hat{x}_o$	Ground truth of edge-sent/network-received usage: $\hat{x}_e \geq \hat{x}_o$
x_e, x_o	The data usage claimed by the edge and operator
$\hat{x}$	Ground truth of the usage to be charged: $\hat{x} = \hat{x}_o + c \cdot (\hat{x}_e - \hat{x}_o)$
x	The negotiated usage in TLC
K_e^+, K_e^-	Edge application vendor's public/private key pair
K_o^+, K_o^-	Cellular operator's public/private key pair
n_e, n_o	Nonces from the edge and operator
s_e, s_o	The edge and operator's message sequence number

to minimize their payment to the operator. We assume the data can be arbitrarily lost from physical to application layers (§3.1). We seek a solution that can

(1) Bridge the data charging gap: It should be resilient to both the data loss and selfish charging;

(2) Be friendly to delay-sensitive edge apps: It should avoid delaying the data transfer whenever possible; and

(3) Support public verifiability: An independent third party (e.g., FCC or court) should be able to verify the charging.

4 INTUITIONS BEHIND TLC

We devise TLC, a **T**rusted and **L**oss-tolerant **C**harging scheme for the 4G/5G cellular edge that achieves all the goals in §3.4. Figure 5 shows TLC overview. Different from the legacy 4G/5G, TLC lets the edge and the operator negotiate the data charging volume. To bridge the gap, TLC's key insight is to *let the data loss and selfish claims cancel out each other*. This process is latency-friendly: It will not block or delay the data transfer within the charging cycle. By the end of the cancellation, a cryptographic Proof-of-Charging (PoC) is constructed by the edge and the operator. The PoC is publicly verifiable by an independent third party (e.g., FCC), without auditing the actual data transfer. TLC can be incrementally deployed with the readily-available mechanisms on the commodity edge nodes and cellular networks. It is scalable with comparable signaling overhead to the legacy 4G/5G. TLC requires the cellular operator and edge app vendor to synchronize their charging start/end time and thus consistent charging cycle T (Table 1). This is achievable via NTP protocol (impact evaluated in §7.2). We next overview the intuitions of TLC's key components. The concrete design will be presented in §5.

Loss-Selfishness Cancellation (§5.1) The primary goal of TLC is to bridge the data charging gap *without* delaying the data transfer. As shown in Theorem 1, this is generally impossible if both the edge and 4G/5G network are honest. Moreover, the charging gap can be caused by not only the data loss, but also the edge/network's selfish charging (§3.3).

To this end, TLC *lets the data loss and selfish charging claims cancel out each other*. Figure 6 illustrates its intuition, and Table 1 lists the notations. TLC lets the edge app vendor and cellular operator negotiate based on their local charging records. For each edge app on each device, it asks the edge app vendor to report the data volume its server (device) has *sent* (denoted as $\hat{x}_e$), and asks the 4G/5G operator to report the data volume the edge device/server has *received* (denoted as $\hat{x}_o$)[4]. This is readily achievable today (§5.4), with incentives to the network and edge (§8). Then for *every* charging cycle, we have $\hat{x}_e \geq \hat{x}_o$ and the data loss as $(\hat{x}_e - \hat{x}_o)$. These assertions hold for all types of the data loss in §3.1 (from physical to application layer). Given this usage pair $(\hat{x}_e, \hat{x}_o)$, the cellular operators may have diverse charging policies: Some may only charge the data that the edge nodes have received (thus $\hat{x}_o$), while others may also charge the lost data $(\hat{x}_e - \hat{x}_o)$ since the operators have consumed their resource for delivery (see [12] for a survey of U.S. and Korean operators' policies). We thus consider the following scheme:

$$\hat{x} = \hat{x}_o + c \cdot (\hat{x}_e - \hat{x}_o), 0 \leq c \leq 1 \text{ and } \hat{x}_o \leq \hat{x}_e \quad (1)$$

where $\hat{x}$ is the final data charging volume, and $c \in [0, 1]$ is a pre-defined charging weight constant for the lost data. We are neutral to the choice of c: It is up to the operator, and should be defined in the data plan and agreed by the operator and edge. $c = 0$ means only the received data will be charged, while $c = 1$ implies all the sent data will be charged. If the edge and operator honestly report $(\hat{x}_e, \hat{x}_o)$, they will stop the negotiation with $\hat{x}$. This scheme does not violate Theorem 1: We do *not* mandate $\hat{x}_e = \hat{x}_o$, thus not delaying the data.

Now consider the interplay between the data loss and selfish charging. Recall that the cellular operators may want to over-charge the edge vendor (thus reporting $x_o \geq \hat{x}_o$), while the edge vendor wants to under-pay for the operators (thus reporting $x_e \leq \hat{x}_e$). However, the data loss poses three constraints over the possible selfish charging claims:

(1) Bounded data charging: The 4G/5G allows an *unbounded* over-charging claim by the operator (§3). Instead, TLC enforces *bounded* charging (Theorem 2 in §5.1): $\hat{x}_o \leq x_e, x_o \leq \hat{x}_e$. The rationale is two-fold. First, the selfish operator (edge) will not under(over)-claim the charging. Second, the operator and edge can *cross-check* each other's record. If the operator's received data is larger than edge's sent data ($x_o > \hat{x}_e$), the edge asserts the operator's claim is selfish and thus rejects this proposal. Similarly, the operator can ensure $x_e \leq \hat{x}_o$.

(2) More selfish charging, less gap: If both the edge and operator are honest, the charging gap is equal to the data loss $\hat{x}_e - \hat{x}_o$. When either is selfish, the charging bound guarantees $\hat{x}_o \leq x_e, x_o \leq \hat{x}_e$. If $x_o \leq x_e$, the charging gap can be *compressed* by the selfish charging: $x_e - x_o \leq \hat{x}_e - \hat{x}_o$. More selfish charging can result in less charging gap by data loss.

(3) Detectable aggressive charging: Once the charging record exhibits $x_e < x_o$, it is asserted that either the edge under-claims or the operator over-claims: In the same charging cycle, the sent data

[4]If multiple edge servers are involved in the same edge app and device, $\hat{x}_e$ is the sum of their received data.

should always be no less than the received data. Upon detecting this signal, a rational edge/operator can take appropriate actions to countermeasure such selfish charging.

With these features, we design a loss-selfishness cancellation game using the minimax theorem in the zero-sum games (§5.1). The game involves the edge app vendor and cellular operators, and guarantees two properties: (1) The charging is *always bounded* by the sent/received data: $\hat{x}_o \le x \le \hat{x}_e$, which is not guaranteed by the legacy 4G/5G; and (2) If both the edge and operator are rational, their optimal charging strategies will result in the expected data charging: $x = \hat{x}$.

Latency Friendliness (§5.2) TLC does not synchronize the edge and networks' charging record. It thus bypasses Theorem 1 and avoids delaying the data in the charging cycle. Moreover, our loss-selfishness cancellation can converge in *1 round*, even if the edge and operator are selfish yet rational.

Public Verifiability (§5.3) Simply bridging the gap is not sufficient to enforce the correct charging. With the legacy network-centric 4G/5G, a selfish cellular operator can ignore the negotiation results and force the edge to pay more than that (e.g., otherwise deactivate the services). TLC thus seeks to regulate the operator's behaviors by making the charging negotiation publicly verifiable. It constructs an unforgeable, undeniable Proof-of-Charging (PoC) using the cryptographic approach. An independent 3rd party (e.g. FCC) can verify the data charging and challenge operator's actions, *without* auditing the data transfer. The operator's over-charging will thus be publicly caught and possibly penalized externally.

Tamper-Resilient Charging Record (§5.4) TLC requires the edge app vendor to report the *sent* data volume, and the network operator to report the *received* volume. TLC achieves this by reusing the readily-available mechanisms in the devices, edge servers, and the 4G/5G infrastructure. But as shown in §5.4, existing mechanisms can suffer from manipulations. For example, a selfish edge vendor can tamper with the operator's charging record and causes under-charging. TLC thus leverages domain-specific, readily-available hardware protections to build tamper-resilient charging records.

5 THE TLC DESIGN

We next elaborate each component in TLC.

5.1 Loss-Selfishness Cancellation

Algorithm 1 elaborates TLC's loss-selfishness cancellation. It is performed by the edge app vendor and cellular operator. They first exchange their charging records (line 4). Then they will decide whether to accept such claims (line 5). If both accept, TLC stops with the charging volume x (line 7–9). If either rejects, the edge and operator should re-claim with a constraint: The next round's reports should be bounded by the range of this round's claim (line 12). Such constraint is visible to both the edge and operator (line 4). The edge (operator) can locally check if the (operator) edge's claim satisfies this constraint in the next round (reject if not enforced).

We next analyze Algorithm 1's effectiveness. We start with the observation that, Algorithm 1 enforces the *bounded data charging*. For comparison, the legacy 4G/5G does not satisfy this property (§3.1): It allows a rational network operator to over-claim arbitrary

Algorithm 1 Loss-Selfishness Cancellation

1: $x_L \leftarrow 0, x_U \leftarrow \infty$; ▷ *(Lower/upper bound of CDRs.)*
2: **while True do**
3: ▷ *(Exchange the CDRs. The order will not affect the result.)*
4: Edge↔Operator: $x_e, x_o \in (x_L, x_U)$;
5: ▷ *(Exchange decisions. The order will not affect the result.)*
6: Edge↔Operator: $R_e, R_o \in \{\texttt{Accept}, \texttt{Reject}\}$;
7: **if** R_e==Accept **and** R_o==Accept **then**
8: $x \leftarrow \begin{cases} x_o + c \cdot (x_e - x_o) & \text{if } x_o \le x_e \\ x_e + c \cdot (x_o - x_e) & \text{otherwise} \end{cases}$
9: Edge & operator sign and store PoC with x (§5.3);
10: **return**; ▷ *(Negotiation completed.)*
11: **else** ▷ *(Otherwise, reclaim CDRs with new bounds.)*
12: $x_L \leftarrow \min\{x_e, x_o\}, x_U \leftarrow \max\{x_e, x_o\}$;
13: **end if**
14: **end while**

data usage for the charging. Specifically, we have the following theorem (proved in Appendix B):

Theorem 2 (Charging bound)**.** *If the edge and operator are rational or honest, Algorithm 1 will stop with* $\hat{x}_o \le x \le \hat{x}_e$.

i.e., the edge will be charged *no more than* its sent data, and *no less than* its received data. As explained in §4, such bound is enforced by edge and network's selfishness *and* cross-check.

We next show that, if both the edge app vendor and cellular operator are rational, their optimal strategy will result in $x = \hat{x}$, thus enforcing the data plan. To understand this, consider their best strategy of claiming the charging. When sending the charging record to the verifier, the edge (operator) may not know the operator (edge)'s claim (line 3)[5]. To minimize its charging, the edge can use the classical *minimax* strategy in the zero-sum game [32]: For each possible charging x_e it claims, the edge computes the *worst-case* charging by considering network's all possible claim x_o. Then the edge reports x_e that minimizes its worst-case charging, i.e.:

$$x_e = \arg\min_{x_e} \max_{x_o | x_e} x \tag{2}$$

Similarly, to maximize its charging, the operator will use the *maximin* strategy to decide its claim:

$$x_o = \arg\max_{x_o} \min_{x_e | x_o} x \tag{3}$$

If $\min_{x_e} \max_{x_o|x_e} x = \max_{x_o} \min_{x_e|x_o} x$, i.e. the operator and edge's optimal strategies are coherent, a *unique* pure-strategy Nash Equilibrium exists, and the game stops at it[6]. We prove that Algorithm 1 meets these criteria (proof in Appendix C):

Theorem 3 (Correctness)**.** *If both the edge and operator are rational, Algorithm 1 will stop with* $x = \hat{x}$.

The proof adopts Von Neumann's minimax theorem [32]. So TLC will provably enforce the data plan for rational edge and operator.

[5] It is possible that the edge waits for operator's report x_o first, and claims x_e based on x_o (the operator may apply similar strategies). But the above minimax strategy still applies, and the proof in Appendix C still holds.

[6] In game theory, a pure strategy deterministically decides a player's action. A two-player zero-sum game has a pure-strategy Nash Equilibrium *if and only if* player 1's minimax strategy equals player 2's maximin strategy [32].

The negotiation will terminate because the operator and edge can achieve their own optimal charging in one round (§5.2); neither the operator or edge will benefit with more rounds (detailed below). Of course, they can use other strategies, but cause sub-optimal charging for their own benefits (tested in §7.1). Note that, if both the operator and edge are honest, the legacy 4G/5G can achieve the same correctness without TLC. However, the legacy 4G/5G cannot prove the honestness of edge/operator, or provide publicly verifiable proofs of charging to the independent third party like TLC (§5.3).

We last consider the potential misbehaviors. A buggy (or misbehaved) operator (edge) may not strictly follow Algorithm 1. It may intentionally reject all claims, insist on untruthful claims, or ignore line 12's constraint and claim larger (smaller) volume (which will be detected by the other and rejected). Both result in longer negotiation. But *neither the operator or edge benefits from such misbehaviors or more rounds.* The operator cannot be paid by the edge before Algorithm 1 stops (no PoC in §5.3), and may even lose this edge customer eventually due to the disruption of network service. The edge cannot continue the network service from the operator before the previous charging cycle is finished. So a rational operator (edge) would avoid misbehaviors.

5.2 Latency Friendliness

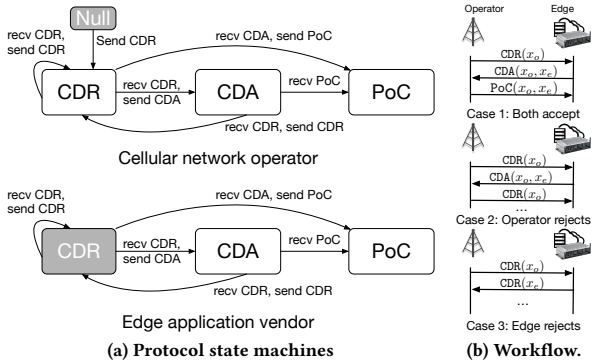
Algorithm 1 is friendly to delay-sensitive edge applications for three reasons. First, within a charging cycle, it will not block the data transfer, or interact with network protocols (e.g., TCP) or operator-side actions. Instead, it only runs at the end of the cycle (e.g., bill cycle stops, or the charging volume exceeds a pre-defined quota). Second, it does not require extra processing or storage overhead for data packets, thus avoiding the processing delay. Last, even at the end of the charging cycle, Algorithm 1 can achieve $x = \hat{x}$ in *1 round only*. This is guaranteed by the following theorem:

Theorem 4 (Latency friendliness)**.** *Algorithm 1 can stop with $x = \hat{x}$ in 1 round if (1) The edge and operator are honest: $x_e = \hat{x}_e, x_o = \hat{x}_o$; or (2) The edge and operator are rational: $x_e = \hat{x}_o, x_o = \hat{x}_e$ (i.e., optimal strategy in Algorithm 1).*

To prove it, one can directly apply ($x_e = \hat{x}_e, x_o = \hat{x}_o$) and ($x_e = \hat{x}_o, x_o = \hat{x}_e$) to Algorithm 1. This is readily achievable in reality: The cellular operator can infer $\hat{x}_e$ with its gateway-based charging functions (§2.1), while the edge app vendor can infer $\hat{x}_o$ using its local traffic monitors (§5.4). Also note Theorem 4 needs both the edge and operator to be honest, *or* both to be rational. If one is honest but the other is rational, Algorithm 1 may converge to $x \neq \hat{x}$. But the bounded charging (Theorem 2) still holds, thus better than legacy 4G/5G's. If either party applies suboptimal strategy or insists on untruthful claims, more rounds may be invoked. But as explained in §5.1, such behaviors hurt both operator (no payment or even customer loss) and edge (delayed or no network service).

5.3 Publicly Verifiable Proof-of-Charging

We next convert Algorithm 1 to a verifiable protocol. The protocol includes two phases: Negotiation and verification. In the negotiation phase, the edge and operator run Algorithm 1 via message exchange. By the end of the negotiation, they construct and store a publicly-verifiable *Proof-of-Charging (PoC)*. In the verification phase, an independent 3rd party accepts PoCs and verifies the charging.



(a) Protocol state machines **(b) Workflow.**

Figure 7: The TLC protocol if the cellular operator initiates the negotiation. Gray states are the initial states.

This enforces the operators and edge to follow the charging from Algorithm 1. The negotiation and verification run at the application layer, without involving 4G/5G signaling traffic.

5.3.1 Setup. The setup runs before the cycle, with 2 steps:
(1) Data plan agreement: The edge and operator first agree on the data plan to be used. The data plan includes the charging cycle time $T = (T_{start}, T_{end})$, and the discounted constant $c \in [0, 1]$, and other information such as pricing, the pre-paid data volume quota, the throttled speed after exceeding the quota, and more (not used in our protocol). The edge nodes and cellular network make (c, T) as public information, and synchronize time based on T (e.g. via NTP protocol [33]);
(2) Cryptographic key setup: TLC requires a public/private key pair for the edge (denoted as K_e^+, K_e^-) and operator (K_o^+, K_o^-). Before the negotiation, the edge and operator publicize K_e^+, K_o^+.

5.3.2 Negotiation and PoC construction. This phase starts by the end of the charging cycle. Either the cellular operator or the edge app vendor initiates the negotiation. Figure 7 illustrates the state machines when the operator initiates the negotiation. The states indicate the sent message, and the state transition occurs if the operator (edge) receives the response from the edge (operator) and sends new messages.

Message types: Three messages are used in Algorithm 1:

• *Charging Data Record (*CDR*).* It reports the usage in the charging cycle (line 3 in Algorithm 1). The edge application vendor and cellular operator's CDR are defined as

$$\text{CDR}_e = \{T, c, s_e, n_e, x_e, \}_{K_e^-}, \text{CDR}_o = \{T, c, s_o, n_o, x_o, \}_{K_o^-}$$

where $s_e(s_o)$ is the edge (operator)'s local sequence number (increased by 1 on sending a message), and $n_e(n_o)$ is a nonce. Compared with the CDRs in 4G/5G, TLC's CDRs are digitally signed by the edge app vendor or operators. The operators can reuse standard 4G/5G CDRs, and sign them as TLC's.

• *Charging Data Acceptance (*CDA*).* Upon receiving the operator (edge)'s CDR, the edge (operator) can decide whether to accept such record (line 5 in Algorithm 1). If yes, the edge (operator) will reply a CDA message. Otherwise, the edge (operator) will implicitly reject and reply a CDR (i.e., reclaiming the charging). The edge and

Algorithm 2 Public Verification by Independent Third Party

Require: PoC, T, c K_v^+, K_e^+
1: $(T', c', x, x_e, x_o, n_e', n_o', s_e, s_o) \leftarrow \texttt{Decrypt}_{K_e^+, K_o^+}(\texttt{PoC})$;
2: **if** T′ ≠T **or** $c' \neq c$ **then**
3: **return false**; ▷ *(Inconsistent data plan)*
4: **end if**
5: **if** $n_e' \neq \texttt{PoC}.n_e$ **or** $n_o' \neq \texttt{PoC}.n_o$ **or** $s_e \neq s_o$ **then**
6: **return false**; ▷ *(Resilient to replay attacks.)*
7: **end if**
8: $x' \leftarrow \begin{cases} x_o + c \cdot (x_e - x_o) & \text{if } x_o \leq x_e \\ x_e + c \cdot (x_o - x_e) & \text{otherwise} \end{cases}$
9: **return** $(x' == x)$;

operator's CDAs are defined as

$$\texttt{CDA}_e = \{T, c, s_e, n_e, x_e, \texttt{CDR}_o\}_{K_e^-}, \texttt{CDA}_o = \{T, c, s_o, n_o, x_o, \texttt{CDR}_e\}_{K_o^-}$$

That is, the operator (edge) copies the edge (operator)'s CDR it receives, and signs it together with its own charging record.

• *Proof of Charging (*PoC*).* On receiving the CDA, the operator (edge) can decide whether to accept the edge (operator)'s record (line 5 in Algorithm 1) and stops the negotiation. If yes, it will construct the PoC based on the edge (operator)'s CDA (line 6–9), reply it to the edge and locally store it as a charging receipt. Otherwise, it will implicitly reject with a CDR (i.e., reclaiming the charging). The PoC is defined as:

$$\text{PoC} = \begin{cases} \{T, c, x, CDA_o\}_{K_e^-} || n_e || n_o & \text{if edge receives CDA} \\ \{T, c, x, CDA_e\}_{K_o^-} || n_e || n_o & \text{if operator receives CDA} \end{cases}$$

where x is the negotiated charging in Algorithm 1 (line 7). Note that, the PoC has been signed by both the cellular operator and the edge app vendor. It is thus an unforgeable, undeniable proof of negotiation for a charging cycle.

Runtime negotiation. Figure 7b shows how the above messages implement Algorithm 1. We assume the cellular operator initiates the process; the edge app vendor can also initiate and follow similar workflow. By the end of the charging cycle, the operator sends its charging record $\texttt{CDR}_o$ to the edge. If the edge accepts, it replies $\texttt{CDA}_e$ to the operator, together with its record x_e. The operator can accept with PoC and stop the negotiation (case 1 in Figure 7b), or reject and re-claim the charging by reinitiating $\texttt{CDR}_o$ (case 2). Similarly, the edge app vendor can reject the operator's claim and repropose the charging record by replying $\texttt{CDA}_o$ (case 3).

5.3.3 Public verification. To prove the charging, the edge (operator) sends (PoC, T, c K_v^+, K_e^+) to a public verifier. Algorithm 2 shows how the verifier works. It first decrypts PoC with edge and operator's public keys. Then it checks if the data plan is coherent with the edge and operator's agreement. To defend the replay attacks with outdated PoCs, the verifier checks if the nonces and sequence numbers are coherent. If so, it further checks if the records (x_e, x_o) and the negotiated volume x are consistent by replaying Algorithm 1. If yes, it confirms that the negotiate succeeds, and $x = \hat{x}$ if both are rational.

5.3.4 Who may serve as a public verifier. In principle, any independent third party can be a public verifier. In reality, the network and edge may have privacy concerns to share their charging records.

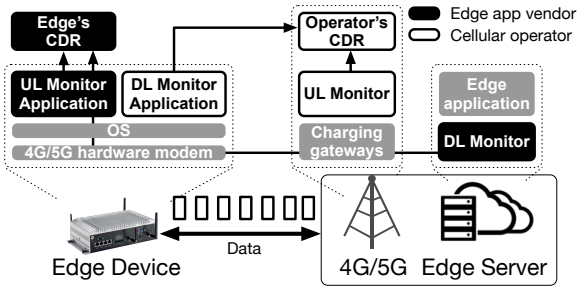


Figure 8: TLC's charging data record collection.

We thus exemplify three realistic settings: (1) *FCC:* As a government agency, FCC is responsible for regulating cellular network behaviors. It can access the edge and operator's charging records via lawful inspections, thus being a verifier; (2) *Courts:* For the lawsuits like [15], the court can verify the charging with the PoC from the edge or operator; (3) *Mobile virtual network operator (MVNO):* The MVNOs (e.g. Google Fi [34] and Aeris [35]) offer cellular services by renting the big operators' infrastructure. When serving an edge app vendor, the MVNOs charge the edge based on the data volume reported by the big operator. They can also get the charging data from edge with their billing mobile apps. In such scenario, the MVNOs may also serve as public verifier to enforce the trusted data volume based on big operator and edge's claims.

5.4 Tamper-Resilient Charging Record

To function correctly, TLC requires the edge app vendor to report the *sent* data volume, and the cellular operator to report the *received* data volume (§5.1). TLC achieves this by reusing the readily-available mechanisms in the edge nodes and cellular networks. It thus simplifies the deployment on the edge and network, and retains comparable signaling overhead to the legacy 4G/5G. Figure 8 shows how TLC achieves this.

• *Uplink traffic (Device⟶Server):* To collect x_e, the edge can count the traffic inside its device-side apps or use `TrafficStat` in Android. To collect x_o, the operator can reuse its charging function at the gateways (P-GW in 4G [20], UPF in 5G [36]);

• *Downlink traffic (Server⟶Device):* To collect x_e, the edge vendor can directly deploy a monitor inside its servers. But to collect x_o, the cellular operator cannot reuse its standard 4G/5G charging function since it's not deployed at the receiver side (i.e., edge device). To this end, we observe that it has been a common practice for the major operators to deploy mobile apps for the user billing report (such as T-Mobile [37] and myAT&T [38]). TLC thus leverages this feature to collect x_o for the operators.

A potential threat is that, such operator's downlink charging record can be tampered by the selfish edge. The operator should deploy its downlink traffic monitor inside the edge devices. A selfish edge device may manipulate the operator's in-device traffic monitor and under-claim the received data x_o. This causes the under-charging and revenue loss for the operators. Note that, other monitors do not suffer from this issue. To address this, consider two strawman options:

Strawman 1: User-space monitor+legacy APIs. Existing mobile OS has APIs for the traffic statistics, such as Android's TrafficStats and Linux's netstat. To collect the downlink traffic, the

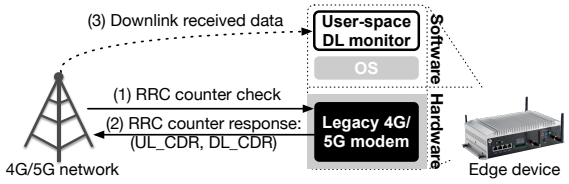


Figure 9: TLC's tamper-resilient downlink monitor.

operator can simply install a user-space mobile app and query the data usage via these APIs. However, this is vulnerable to manipulation. A selfish edge can modify these APIs to report less data, which can be realized with customized Linux or Android images (e.g. AOSP [39]).

Strawman 2: System monitor with root privilege. An alternative option for the operator is to install a system app and directly inspect the traffic (rather than query the legacy APIs). This approach is also the *de facto* solution for the Voice-over-LTE [40], the high-quality call service in 4G. It is resilient to the edge's manipulation: The operator can observe every packet the device consumes. But this solution requires the system privilege (e.g., root in Android and Linux), and raises privacy concerns for the edge devices.

Our solution: User-space monitor+hardware protection. We devise a domain-specific, readily-available approach that offers hardware-level resilience *without* system privilege. We note that, 4G/5G defines a standard mechanism for the operators to query the device-received traffic: The RRC COUNTER CHECK procedure in the Radio Resource Control (RRC) protocol [41, 42]. It is readily available in the commodity devices and 4G/5G base stations. Similar to most in-device 4G/5G mechanisms, this procedure is implemented in the hardware, thus resilient to selfish edge's manipulation[7]. Figure 9 shows how it works. To send/receive the data, the device should establish a radio connection to the base station. The base station can send RRC COUNTER CHECK to the device, and query the device-received traffic over this connection. The modem will reply its usage in RRC COUNTER CHECK RESPONSE. As long as the hardware modem is trusted and tamper-resilient, the operator obtains trusted records. Moreover, the operator still retains its control of downlink traffic statistics since it initiates the counter check. Note that, activating the readily-available RRC COUNTER CHECK function is straightforward and acceptable to the operator: It is similar to other configurations performed by the radio engineers, and does not require upgrading the base station's software or hardware. These properties provide more technical incentives for the operator to adopt our solution.

To this end, TLC leverages RRC COUNTER CHECK to build the operator's downlink traffic monitor. As shown in Figure 9, the operator installs a user-space monitor on the device (no root), and activates RRC COUNTER CHECK in its base stations. Before the base station disconnects the device[8], the base station initiates the RRC COUNTER CHECK, queries device's received data volume from the hardware, and sends this usage back to the operator's app. The operator's app aggregates the usage from all radio connections as its downlink record x_o. In this way, the additional RRC COUNTER CHECK messages invoked by TLC will be bounded by the number of RRC connection releases.

[7]Some tools (e.g. QXDM [43] and MobileInsight [44]) can read the modem's internal states, but cannot change the modem's traffic statistics. We are unaware of attacks that can manipulate the cellular hardware modem.

[8]In 4G/5G, each radio connection release is initiated by the base station (by sending RRC CONNECTION RELEASE message) when it observes no data transfer from/to the device [41, 42].

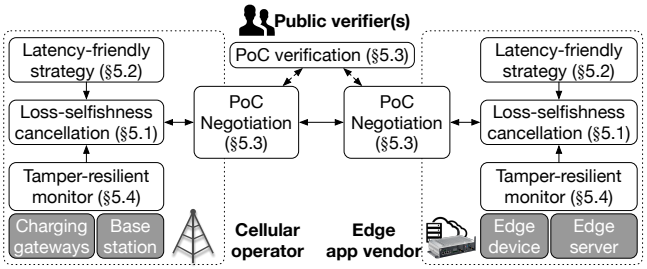


Figure 10: TLC implementation overview.

6 SYSTEM IMPLEMENTATION

Figure 10 overviews TLC's implementation. We realize it as an Java applet on the edge, and an extended policy of LTE offline charging functions (OFCS) in the cellular network.

Cellular operator: TLC is realized atop existing charging functions (CDF in 4G [6], CHF in 5G [7]) with four modules:

• *Loss-selfishness cancellation (§5.1):* It is a post-processing logic of charging records in OFCS (similar to policies in §2.1). It accepts the charging records, decides (x_o, R_o) with operator's strategy (detailed below), and notifies TLC protocol.

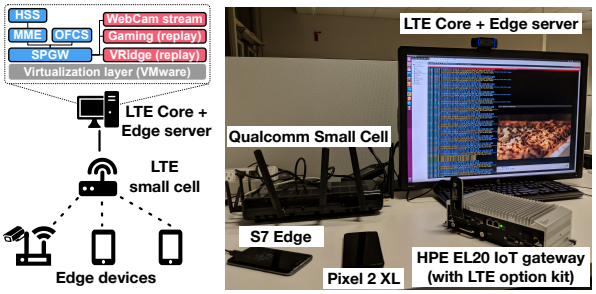
• *Latency-friendly negotiation strategy (§5.2):* To mitigate the negotiation rounds, we follow Theorem 2 and 4: On receiving the edge's claim x_e, the cellular operator checks if $x_e < \hat{x}_o$ and rejects if such condition holds. Meanwhile, as a rational operator, it always claims $x_o = \hat{x}_e$ using the maximin strategy in §5.1. This helps achieve 1-round negotiation (Theorem 4).

• *Publicly-verifiable negotiation (§5.3):* It is realized as an app-layer protocol. It runs the state machine in Figure 7a, accepts the CDR/CDA/PoC from the edge app vendor, and replies the signed messages. We adopt the `java.security` for the key-pair generation (RSA-1024) and encryption/decryption.

• *Tamper-resilient charging records (§5.4):* TLC activates the base stations' RRC COUNTER CHECK for the downlink records (§5.4), and reuses the readily-available uplink records from the 4G/5G gateways. If RRC COUNTER CHECK is not activated, the operator can still roll back to the device APIs or system monitor, at the cost of inaccuracy or privacy (§5.4).

Edge app vendor: We realize TLC as a Java applet on Linux, and APK on Android. It collects the uplink records from the device via `TrafficState` on Android, and downlink records from server via `/proc/EDGE_APP_PID/net/netstat` on Linux. The edge server realizes the loss-selfishness cancellation (§5.1) and minimax latency-friendly strategy (§5.2). At runtime, the device runs TLC protocol in Figure 7a, and uses `java.security` for cryptographic actions.

Public verifier(s): We implement TLC's public verification as a standalone Java applet. The verification is performed based on edge or cellular operator's requests. It accepts the data plan parameters (c, T), the Proof-of-Charging (PoC) from the edge app vendor or the cellular operator, and their public keys. Then the TLC binary

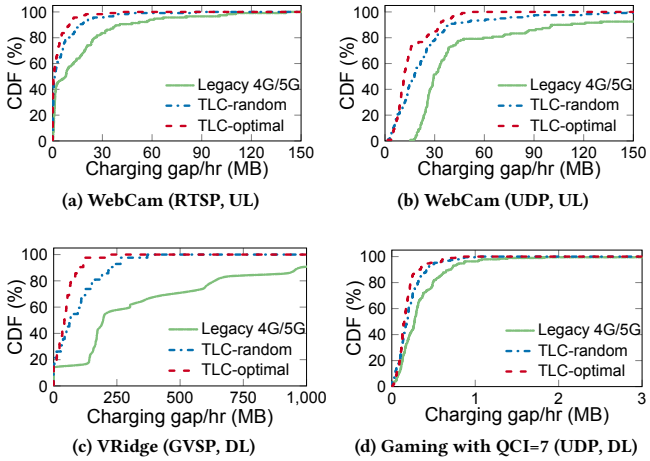


(a) System setup (b) Testbed

	WebCam stream	Online gaming	VRidge
# Charging data record	914,565	58,903	31,448
Charging data volume	171.6 GB	314.0MB	112.5GB

(c) Experimental dataset

Figure 11: Our experimental setup.



(a) WebCam (RTSP, UL) (b) WebCam (UDP, UL)

(c) VRidge (GVSP, DL) (d) Gaming with QCI=7 (UDP, DL)

Figure 12: Overall charging gap ($c = 0.5$).

	Avg. Bitrate (Mbps)	Legacy 4G/5G $\Delta = \lvert x - \hat{x}\rvert$ (MB/hr)	Legacy 4G/5G $\epsilon = \Delta/\hat{x}$	TLC-optimal $\Delta = \lvert x - \hat{x}\rvert$ (MB/hr)	TLC-optimal ϵ	TLC-random $\Delta = \lvert x - \hat{x}\rvert$ (MB/hr)	TLC-random ϵ
WebCam (RTSP)	0.77	16.56	17.0%	3.27	2.2%	6.02	5.1%
WebCam (UDP)	1.73	54.68	8.1%	15.59	2.0%	23.72	3.3%
VRidge (Portal 2)	9.0	384.49	21.9%	48.07	1.8%	93.3	4.5%
Gaming w/ QCI=7	0.02	0.34	3.2%	0.18	1.6%	0.21	1.9%

Table 2: Average charging gap ($c = 0.5$).

runs Algorithm 2 to decide whether the charging is coherent with the negotiation.

7 EVALUATION

We assess TLC's effectiveness of bridging the charging gap (§7.1), and examine its system efficiency and overhead (§7.2).

Ethical evaluation: This work does not raise any ethical issues. We understand that some experiments of charging might be detrimental to the operators or users. So we run the experiments in a responsible manner. Instead of operational 4G/5G, we run experiments in a testbed with carrier-grade LTE core and small cells. Besides, in the tests beneficial to operators or edge, we test strictly in a controlled setting.

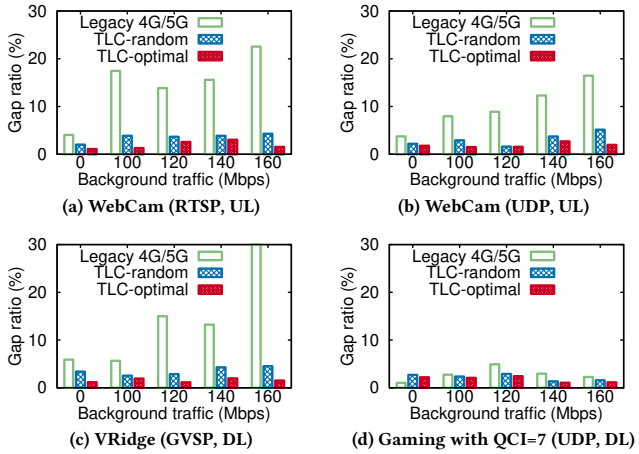
Experimental setup: Figure 11 shows our testbed. It consists of an experimental LTE network (with OpenEPC [16] as LTE core and a Qualcomm small cell), an edge server (HP Z840 Workstation with 2.3GHz 10-core Intel Xeon E5-2650v3 CPU, 32GB RAM, 1TB SSD, Logitech HD Pro Webcam C920, and Ubuntu 16.04), and three edge devices (HPE EL20 IoT gateway, Samsung S7 Edge, and Google Pixel 2 XL). In LTE, the small cell transmits over LTE FDD band 2, with 20MHz carrier bandwidth and EARFCN=900 (we are authorized to use this spectrum by FCC for the indoor tests). This setting is similar to the U.S. operators' commercial band deployments. The LTE core is deployed in the server as virtual machines (using VMware Workstation 15), each running as a function node (offline charging system (OFCS), charging gateways (SPGW), policy/charging functions (PCRF), mobility management entity (MME), and home subscriber server (HSS)). The edge server is co-located with LTE core, and connects to the small cell via 1Gbps Ethernet. The devices communicate with the server via LTE, and run edge apps or background traffic. TLC runs on the edge nodes and the LTE core network.

7.1 Overall Effectiveness

We evaluate TLC's charging gap reduction by emulating three edge scenarios in §2.2: (1) *WebCam stream for video analytics:* We run VLC [45] to stream the edge device (server)'s runtime camera frames to server (device). The frames are encoded in H.264 (1920×1080p, 30FPS), and streamed by RTSP and legacy UDP; (2) *Edge-based VR:* We replay the `tcpdump` logs from [28]. These logs are from edge-based VRidge [26] and Portal 2 over U.S. LTE operators. The VR graphical frames were encoded in 1920×1080p 60FPS, and streamed via GVSP protocol. We replay the packets (via `tcprelay`) from the edge server to the device. (3) *Online gaming acceleration:* We collect a 1-hour `tcpdump` trace of *King of Glory*, a Tencent's multi-player online game with 200M monthly active players in China [46]. We replay this trace (via `tcprelay`) from the edge server to the device. To emulate the acceleration, we assign QCI=7 (for interactive games) to the game traffic in the LTE core (for comparison, the background LTE traffic has QCI=9 as lowest priority).

We compare TLC with the state-of-art charging schemes. Three schemes are tested: (1) **(Honest) legacy 4G/5G**. It is the baseline. The operator *honestly* charges the user by gateway's CDRs (§2.1); (2) **TLC-optimal**. It follows the optimal strategy in §5.1, and achieves negotiation in 1 round. This assumes both operator and edge app vendor are rational; (3) **TLC-random**. It assesses the case when the operator and edge app vendor are selfish but unaware of the optimal strategy. In each round, the operator (edge) will seek to over-claim (under-claim) its charging volume (not necessarily the optimal one in TLC-optimal). It thus uniformly chooses the charging volume larger (smaller) than $\hat{x}_o$ ($\hat{x}_e$) and run the negotiation. For all these strategies, we run the edge apps and record the data usage on the edge and network every 1s. Each round takes 1 hour (i.e. the charging cycle T=1hr). After that, we run above schemes, and compute their absolute charging gap $\Delta = |x - \hat{x}|$ and relative gap ratio $\epsilon = \Delta/\hat{x}$. We repeat the experiments with various congestion (with [0, 1Gbps] iperf UDP background traffic to a separate phone (S7 Edge)), wireless intermittent disconnectivity levels (with [-95dBm, -120dBm] signal strength), and charging plan parameters $c \in [0, 0.25, 0.5, 0.75, 1]$. Figure 11c summarizes the dataset size.

Figure 12 and Table 2 show the gap reduction. Figure 13 and Figure 14 compare TLC charging gap reduction under various congestion and intermittent wireless connectivity levels ($c = 0.5$). Figure 15 plots the impact of loss weight c.



(a) WebCam (RTSP, UL) (b) WebCam (UDP, UL)

(c) VRidge (GVSP, DL) (d) Gaming with QCI=7 (UDP, DL)

Figure 13: Charging gap under congestion.

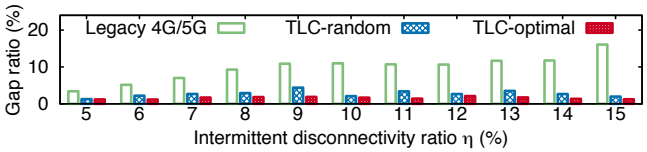


Figure 14: Charging gap in intermittent connectivity.

Overall charging gap reduction. Figure 12 and Table 2 show that, TLC reduces the gaps for all the tested scenarios. Compared with the legacy 4G/5G, TLC-optimal reduces the average absolute gap Δ by 80.2% in RTSP WebCam streaming (16.56⟶3.27MB/hr), 71.5% in legacy UDP WebCam streaming (54.68⟶23.72MB/hr), 87.5% in edge-based VR using VRidge (384.49⟶48.07MB/hr), and 47.06% in the online gaming (0.34 ⟶ 0.18 MB/hr). The average relative gap ratio ϵ is also reduced (17.0%⟶2.2% in RTSP streaming, 8.1%⟶2.0% in UDP streaming, 21.9%⟶1.8% in VR, and 3.2%⟶1.6% in online gaming). The application with more traffic (e.g., VR) benefits more from TLC, which is more vulnerable to the data loss. Our current TLC implementation does not fully close the gap due to the errors in the charging records (assessed in §7.2). But even so, TLC-optimal achieves ≤2.5% relative gap ratio ϵ.

Comparison of TLC's strategies. Figure 12 and Table 2 show that, both TLC-optimal and TLC-random reduce the gaps. TLC-optimal saves more due to its provable optimality.

Network congestion and QoS's impact. Figure 13 shows that, all the tested scenarios benefit more from TLC with more congestion. The reason is that, the legacy 4G/5G experiences larger loss-induced gap in congestion, while TLC-optimal retains optimality regardless of the congestion levels. Note higher QoS priority could help mitigate loss from congestion (gaming in Figure 12d). The charging gap is negligible in legacy 4G/5G, and TLC still helps further reduce it.

Intermittent wireless connectivity's impact. We define the intermittent disconnectivity ratio $\eta = t_{disconn}/t_{total}$, where t_{total} is the total elapsed time in each experiment round, and $t_{disconn}$ is the duration that the device loses uplink and downlink connectivity. Figure 14 compares the charging gaps under various η in UDP-based WebCam streaming; other applications have similar results. It shows that, TLC reduces more gaps with heavier intermittent connectivity levels, which results in larger gaps in the legacy 4G/5G.

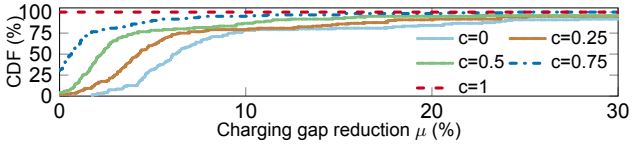
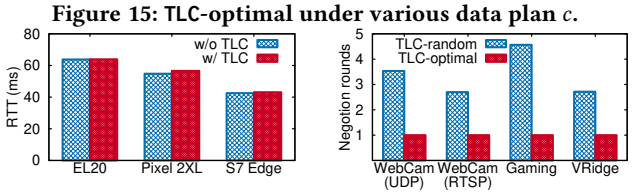


Figure 15: TLC-optimal under various data plan c.



(a) RTT within charging cycle (b) Negotiation after charging cycle

Figure 16: TLC's impact on data latency.

Data plan parameter c's impact. As defined in §4, the discount factor c decides the charging weight for the lost data. Figure 15 compares TLC-optimal's charging gap reduction ratio over legacy 4G/5G: $\mu = (x_{legacy} - x_{\text{TLC}})/x_{legacy}$. It shows that, smaller c results in more gap reductions in TLC: With less charging weights for the lost data, the legacy 4G/5G tends to have larger gaps. When $c = 1$ (i.e. all the lost data will be charged), TLC is the same as the *honest* legacy 4G/5G. But it still helps avoid gaps due to the selfish charging.

7.2 Efficiency and Overhead

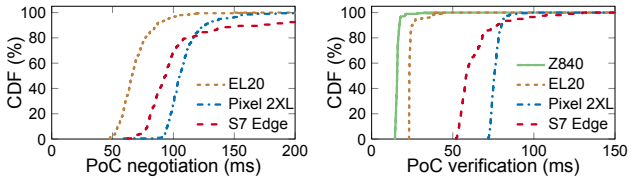
Latency friendliness: We examine TLC's impact on the data latency in two dimensions. First, within the charging cycle, Figure 16a plots the average round-trip time (ping 200 rounds for each). For all the tested edge devices, RTT exhibits marginal differences with/without TLC. As explained in §5.2, TLC does not block the data transfer in the charging cycle.

Second, by the end of the charging cycle, we assess the negotiation rounds needed by TLC in Algorithm 1. Figure 16b shows that, TLC-optimal converges in 1 round for all tested edge applications (Theorem 4). Instead, TLC-random needs 3.5 rounds for UDP-based WebCam, 2.7 for RTSP-based WebCam, 4.6 for online gaming, and 2.7 for VR on average.

Negotiation and public verification: We next assess the scalability and overhead of TLC protocol in §5.3. For each experiment round in §7.1, we record its elapsed time and PoC. We then compute the elapsed time of verifying a PoC.

Figure 17 shows the result with TLC-optimal. To negotiate a PoC, it takes 65.8ms, 105.5ms, and 93.7ms on average on HPE EL20, Google Pixel 2 XL, and Samsung S7 Edge, respectively. The negotiation time mainly includes the cryptographic computation (contributing 54.9% on average), and the round-trip between device and network (45.1%). It can be reduced by shortening the round-trip time (e.g. negotiation between edge server and network). The storage overhead is also marginal: Each PoC consumes 796 bytes for the edge app vendor and cellular operator (most being padding bits in RSA-1024 and thus compressable). To verify a PoC, it takes 23.2ms, 75.6ms, 58.3ms, and 15.7ms on average on HPE EL20, Google Pixel 2 XL, Samsung S7 Edge, and HP Z840. That is, a verifier (e.g., FCC) with a single HP Z840 workstation can process 230K verification requests per hour on average.

Tamper-resilient charging record: We last quantify the accuracy of TLC's tamper-resilient record in §5.4. For the uplink, TLC



LTE CDR	TLC CDR	TLC CDA	TLC PoC	Total signaling overhead
34 bytes	199 bytes	398 bytes	796 bytes	1,393 bytes, 3 msg

Figure 17: Proof-of-Charging's cost (TLC-optimal).

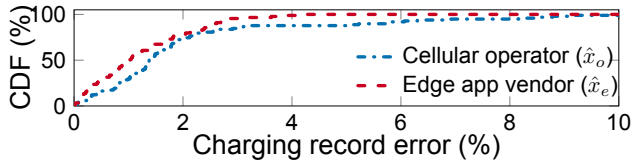


Figure 18: The accuracy of TLC's tamper-resilient CDR.

achieves 100% accuracy since the edge app vendor and cellular operator reuse their mechanisms. For the downlink, the operator uses RRC COUNTER REQUEST instead of gateways. The edge counts the data it initiates rather than what the gateway receives. Both may incur inaccuracies.

Figure 18 plots the error ratio for TLC's downlink records. We confirm that the errors are marginal. For the operator's record, the error is the gap between gateway-based and RRC COUNTER REQUEST-based charging: $\gamma_o = |\hat{x}_o^{rrc} - \hat{x}_o|/\hat{x}_o$. The average error is 2.0%, and 95% of records have ≤7.7% error. The maximum error observed is 12.7%, which is due to the asynchronous charging cycle start/end. For the edge server, the error is the gap between the gateway-based charging and edge's monitoring: $\gamma_e = |\hat{x}_e^{gw} - \hat{x}_e|/\hat{x}_e$ The average error is 1.2%, and 95% of records have ≤2.9% error. The maximum error observed is 4.3%. Such errors are due to the asynchronous charging cycle between edge and network, and can be reduced with time synchronizations (e.g., via NTP [33] or 4G/5G physical-layer synchronization signals [47]).

8 DISCUSSION

Economic deployment incentives: Besides the technical feasibility (§4–5), both the edge app vendor and cellular operator have the economic incentives to deploy TLC. The edge deploys TLC to prevent the over-charging: Otherwise, the cellular operators will roll back to legacy 4G/5G, which allows unbounded selfish charging (§3.4). The operator is motivated via *competition*: An operator with TLC will gain the unique competitive edge (i.e., trusted charging) over other operators without TLC, and attract more users (revenue). For example, if operator A deploys TLC but operator B does not, B's user may switch to A to avoid over-billing and thus lead to B's revenue loss. This strategy is effective for the prepaid edge/IoT users or MVNOs, whose monthly user churn rate can be up to 25% [48].

Multi-access edge: Some edge scenarios (e.g. self-driving and vehicular communication [1]) combine multiple operators' 4G/5G to improve coverage. TLC can be extended to this scenario: For each 4G/5G operator, the edge nodes run TLC to negotiate the per-operator charging. The edge device should install each operator's tamper-resilient monitor in §5.4 (a common practice for commodity phones). To avoid the interference, the edge should classify its data traffic by operators when generating the charging records.

Generic mobile data charging: TLC is currently specific to the edge. Its core idea can be extended to the generic mobile data charging between operators and individual users. In this case, TLC is readily applicable to the uplink traffic. For the downlink, TLC may have over-charging since the data can be lost from Internet to the 4G/5G core. But such over charging is still bounded by the lost data from Internet to cellular gateways (detailed in Appendix D), thus outperforming legacy 4G/5G. We will enhance the downlink support in the future work.

9 RELATED WORK

The cellular edge and 5G are being actively standardized [1, 7, 8, 17, 42] and deployed [4]. TLC complement these efforts; we are unaware of any standards or discussions that seek to enforce the trusted data charging in edge or 5G. In the area of mobile data charging, prior research [9–12] has found various charging gaps by the data loss in 3G/4G (classified in §3.1). Instead, we study the gap from the data loss and selfish charging in the edge, and devise a solution that bridges the gap, is latency-friendly, and publicly verifiable.

Bridging the data charging gap is a sub-category of the verifiable resource accounting. Similar problems have been studied in other contexts, such as the outsourced cloud computing [49, 50] and mobile ad-hoc routing [51, 52]. Instead, we focus on a unique context (cellular edge) and resource (network data transfer). Our work also differs from the general accountable IP protocols [53, 54]: TLC focuses on the charging, and does not incur per-packet overhead or delays. TLC can also be viewed as a game mechanism design (or "inverse game theory" [32]) with cryptographic techniques. Specifically, it is inspired by the bargaining theory [55, 56], but generalizes this model from the economics to the cellular edge setting.

10 CONCLUSION

The cellular edge promises "anywhere, anytime" low-latency services for emergent applications. Ideally, it should enforce trusted charging: the cellular operator and edge app vendor should agree on the charged data usage. Unfortunately, the data charging gap challenges this expectation in the legacy 4G/5G. TLC thus adopts game theory to bridge the gap without delaying the traffic, and leverages the cryptographic tools to build the publicly verifiable Proof-of-Charging.

In a broader context, TLC can be viewed as an initial step toward enhancing the trust between the cellular operators and edge app vendors. The 4G/5G cellular networks largely follow the operator-centric design, leaving limited transparency to their users. Without cooperation, it is hard for the cellular operators and users to fully trust each other. We wish TLC could call the academia and industry's attention for this issue, and stimulate more research and standardization efforts toward trusted, verifiable cellular edge services.

ACKNOWLEDGEMENTS

We greatly thank our shepherd, Dr. Nikolaos Laoutaris, and the anonymous reviewers for their constructive comments.

REFERENCES

[1] ETSI. GS MEC 022: Multi-access Edge Computing (MEC); Study on MEC Support for V2X Use Cases, Sep. 2018.
[2] MIT Technology Review. Moscow Billboard Targets Ads Based on the Car You are Driving. https://www.technologyreview.com/s/603743/moscow-billboard-targets-ads-based-on-the-car-youre-driving/, Mar. 2017.
[3] Tencent. Intelligent Network Optimization. https://cloud.tencent.com/product/ino, 2018.
[4] Envrmnt (by Verizon). Cloud XR Experience on 5G with Mobile Edge Networks. https://www.slideshare.net/AugmentedWorldExpo/raheel-khalid-envrmnt-by-verizon-cloud-xr-experience-on-5g-with-mobile-edge-networks, Jun. 2018.
[5] STL partners and Intel and HPE. Edge Computing: The Telecom Business Models. https://builders.intel.com/docs/networkbuilders/edge-computing-the-telco-business-models.pdf, Oct. 2017.
[6] 3GPP. TS23.401: General Packet Radio Service enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access, Dec. 2018.
[7] 3GPP. TS23.501: System Architecture for the 5G System, Dec. 2018.
[8] ETSI. GS MEC 002: Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements, Oct. 2018.
[9] Chunyi Peng, Guanhua Tu, Chiyu Li, and Songwu Lu. Can We Pay for What We Get in 3G Data Access? In *MobiCom*, Aug. 2012.
[10] GuanHua Tu, Chunyi Peng, ChiYu Li, Xingyu Ma, Hongyi Wang, Tao Wang, and Songwu Lu. Accounting for Roaming Users on Mobile Data Access: Issues and Root Causes. In *MobiSys*, Jun. 2013.
[11] Chunyi Peng, Chi-Yu Li, Hongyi Wang, Guan-Hua Tu, , and Songwu Lu. Real Threats to Your Mobile Data Bills. In *ACM CCS*, Nov 2014.
[12] Younghwan Go, Jongil Won, Denis Foo Kune, EunYoung Jeong, Yongdae Kim, and KyoungSoo Park. Gaining Control of Cellular Traffic Accounting by Spurious TCP Retransmission. In *Network and Distributed System Security (NDSS) Symposium 2014*. Internet Society, 2014.
[13] Google Product Forums. Over-billing, kinda like Apple throttling. https://productforums.google.com/forum/#!msg/project-fi/nhuNE6YqbWQ/YcMgNz9UDAAJ, Jan 2018.
[14] AT&T. Excessive Data Usage. https://forums.att.com/t5/Wireless-Account/Excessive-data-usage/td-p/5173106, 2017.
[15] AndroidAuthority. Lawsuit alleges Project Fi charges customers for Wi-Fi use. https://www.androidauthority.com/lawsuit-alleges-project-fi-charges-customers-wifi-use-835973/, Feb 2018.
[16] OpenEPC. https://www.corenetdynamics.com/products, 2019.
[17] ETSI. GS MEC 003: Mobile Edge Computing (MEC); Framework and Reference Architecture, Oct. 2018.
[18] AT&T. Unlimited mobile data plan. https://www.att.com/plans/unlimited-data-plans.html, 2018.
[19] Tencent. Intelligent Network Optimization: SDK Manual. https://main.qcloudimg.com/raw/document/product/pdf/594_11589_cn.pdf, 2018.
[20] 3GPP. TS23.203: Policy and Charging Control Architecture, Mar. 2015.
[21] ETSI. GS MEC 012: Mobile Edge Computing (MEC); Radio Network Information API, Jul. 2017.
[22] ETSI. GS MEC 015: Mobile Edge Computing (MEC); Bandwidth Management API, Oct. 2017.
[23] Junchen Jiang, Rajdeep Das, Ganesh Ananthanarayanan, Philip A Chou, Venkata Padmanabhan, Vyas Sekar, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, Renat Vafin, and Hui Zhang. Via: Improving Internet Telephony Call Quality Using Predictive Relay Selection. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 286–299. ACM, 2016.
[24] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*, pages 393–406, 2017.
[25] Tencent Instutite of Games. Mobile Gaming Server: UDP or TCP? (in Chinese). http://gad.qq.com/article/detail/21479, Aug. 2016.
[26] VRidge. RiftCat. https://riftcat.com/vridge, 2018.
[27] Wikipedia. GigE Vision stream protocol (GVSP). https://en.wikipedia.org/wiki/GigE_Vision, 2017.
[28] Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, and Songwu Lu. Enabling Mobile VR in LTE Networks: How Close Are We? In *The 44th ACM Annual Conference of Special Interest Group on Measurement and Evaluation (SIGMETRICS'18)*, Irvine, California, USA, June 2018. Dataset at https://ucla.app.box.com/s/cwd6yt8tg5orhbltckkvho9s73crk74d.
[29] 3GPP. TS24.008: Core Network Protocols; Stage 3, Jun. 2014.
[30] Seth Gilbert and Nancy Lynch. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services. *ACM SIGACT News*, 33(2):51–59, 2002.
[31] Mrinal Saha. How to Clear Data Usage on Android. https://techwiser.com/clear-data-usage-on-android/, Jan 2016.
[32] Gibbons, Robert. *Game Theory for Applied Economists*. Princeton University Press, 1992.
[33] Network Time Protocol (NTP). https://en.wikipedia.org/wiki/Network_Time_Protocol, 2018.
[34] Google. Google Fi. https://fi.google.com/, 2018.
[35] Aeris. https://www.aeris.com/, 2018.
[36] 3GPP. TS32.291: Charging Management; 5G System, Charging Service; Stage 3, Sep 2018.
[37] T-Mobile bill app. T-Mobile. https://www.t-mobile.com/apps/download-t-mobile-app, 2018.
[38] myAT&T app. AT&T. https://www.att.com/my/, 2018.
[39] Google. Android Open Source Project. https://source.android.com/, 2019.
[40] Chi-Yu Li, Guan-Hua Tu, Chunyi Peng, Zengwen Yuan, Yuanjie Li, Songwu Lu, and Xinbing Wang. Insecurity of Voice Solution VoLTE in LTE Mobile Networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 316–327. ACM, 2015.
[41] 3GPP. TS36.331: E-UTRA; Radio Resource Control (RRC), Sep. 2018.
[42] 3GPP. TS38.331: 5G NR; Radio Resource Control (RRC), Sep. 2018.
[43] Qualcomm. QxDM Professional - QUALCOMM eXtensible Diagnostic Monitor. http://www.qualcomm.com/media/documents/tags/qxdm.
[44] MobileInsight. http://www.mobileinsight.net, 2017.
[45] VLC media player. https://www.videolan.org/, 2019.
[46] King of Glory. Wikipedia. https://en.wikipedia.org/wiki/Wangzhe_Rongyao, 2018.
[47] 3GPP. TS36.211: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation, 2017.
[48] Martyn Warwick. It's nothing personal - and that's the trouble for MVNOs. https://www.telecomtv.com/content/telco-and-csp/its-nothing-personal-and-thats-the-trouble-for-mvnos-16147/, Nov 2017.
[49] Vyas Sekar and Petros Maniatis. Verifiable Resource Accounting for Cloud Computing Services. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 21–26. ACM, 2011.
[50] Chen Chen, Petros Maniatis, Adrian Perrig, Amit Vasudevan, and Vyas Sekar. Towards Verifiable Resource Accounting for Outsourced Computation. In *ACM SIGPLAN Notices*, volume 48. ACM, 2013.
[51] Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1987–1997. IEEE, 2003.
[52] Sheng Zhong, Li Erran Li, Yanbin Grace Liu, and Yang Richard Yang. On Designing Incentive-Compatible Routing and Forwarding Protocols in Wireless Ad-Hoc Networks. *Wireless Networks*, 13(6):799–816, 2007.
[53] Andersen, David G and Balakrishnan, Hari and Feamster, Nick and Koponen, Teemu and Moon, Daekyeong and Shenker, Scott. Accountable Internet Protocol (AIP). In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 339–350. ACM, 2008.
[54] David Naylor, Matthew K Mukerjee, and Peter Steenkiste. Balancing Accountability and Privacy in the Network. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 75–86. ACM, 2014.
[55] Rubinstein, Ariel. Perfect Equilibrium in a Bargaining Model. *Econometrica: Journal of the Econometric Society*, pages 97–109, 1982.
[56] John Nash. Two-Person Cooperative Games. *Econometrica: Journal of the Econometric Society*, pages 128–140, 1953.

Appendices are supporting material that has not been peer reviewed.

A PROOF OF THEOREM 1

Proof. We prove it by contradiction in a worst-case scenario. Assume all the data packets from the network to device are lost during the transfer (e.g. the device stays in a dead zone). Note that the network may still increase its local counter $\hat{x}_o$ based on these traffic. In the presence of loss, no matter how long the edge device waits, it is impossible for it to differentiate the following two cases: (1) The network has updated the traffic counter as $\hat{x}_o$; (2) The network has updated the traffic counter as $\hat{x}'_o$ ($\neq \hat{x}_o$). Therefore, the device cannot decide whether to update its local traffic counter $\hat{x}_e$ as $\hat{x}_o$ or $\hat{x}'_o$. It either responds the charging query under the inconsistent states (thus incurring charging gaps $\hat{x}_o \neq \hat{x}_e$), or suspends (delays) the query until all lost data is recovered (thus violating the eventual query availability). □

B PROOF OF THEOREM 2

Proof. We first prove that, when Algorithm 1 stops, the charging report is always bounded:

$$\hat{x}_o \le x_e \le \hat{x}_e \text{ and } \hat{x}_o \le x_o \le \hat{x}_e$$

A honest edge (operator) will always report $x_e = \hat{x}_e$ ($x_o = \hat{x}_o$), thus satisfying this bound. So we focus on the rational edge and operator. First note that, the negotiated charging x (line 8 in Algorithm 1) is positively monotonic with x_e and x_o ($0 \le c \le 1$). So the rational edge will not over-claim its charging, and the rational operator will not under-claim its charging. So we have

$$x_e \le \hat{x}_e \text{ and } x_o \ge \hat{x}_o$$

We next prove $x_e \ge \hat{x}_o$; $x_o \le \hat{x}_e$ follows similar proof. The operator can immediately detect $x_e < \hat{x}_o$ based on its local ground truth $\hat{x}_o$. Now the operator should choose to accept or reject the conflict (line 6):

• *Option A: Reject when $x_e < \hat{x}_o$.* Then Algorithm 1 will re-initiate and re-claim (line 16). If so, Algorithm 1 can only stop if $x_e \ge \hat{x}_o$;

• *Option B: Accept when $x_e < \hat{x}_o$.* If the edge rejects, Algorithm 1 will re-initiate and re-claim (same as Option A). If the edge also accepts, Algorithm 1 will stop with $x_e < \hat{x}_o$ and x. But since larger x_e results in larger x, Option B will result in less charging than Option A for the operator.

So to maximize its charging, a rational operator should always reject when $x_e < \hat{x}_o$. Therefore Algorithm 1 will stop with $\hat{x}_o \le x_e \le \hat{x}_e$. Similar proof can show it will stop with $\hat{x}_o \le x_o \le \hat{x}_e$.

Then given $\hat{x}_o \le x_e \le \hat{x}_e$ and $\hat{x}_o \le x_o \le \hat{x}_e$, we immediately have $\hat{x}_o \le x \le \hat{x}_e$ according to line 8 in Algorithm 1 (note $0 \le c \le 1$ as defined in Table 1). □

C PROOF OF THEOREM 3

Proof. Consider the edge's strategy of reporting x_e. As shown in §5.1, the optimal approach for the edge is the minimax strategy: Given each x_e, it first computes the worst-case charging given any possible x_o (line 8 in Algorithm 1):

$$\max_{x_o | x_e} x = \max_{x_o} \begin{cases} x_o + c \cdot (x_e - x_o) & \text{for } x_o \le x_e \\ x_e + c \cdot (x_o - x_e), & \text{for } x_o > x_e \end{cases} \tag{4}$$

When $x_o \le x_e$, we have

$$\max_{x_o \le x_e} \{x_o + c \cdot (x_e - x_o)\} = x_e + c \cdot (x_e - x_e) = x_e$$

as $0 \le c \le 1$. If $x_o > x_e$, Theorem 2 asserts $x_o \le \hat{x}_e$, $x_e \le \hat{x}_e$. So

$$\max_{x_o > x_e} \{x_e + c \cdot (x_o - x_e)\} = (1 - c) \cdot x_e + c \cdot \hat{x}_e > x_e$$

So $\max_{x_o | x_e} x = (1 - c) \cdot x_e + c \cdot \hat{x}_e$. Then the edge will decide its report x_e to minimize the worst-case charging, so we have

$$\min_{x_e} \max_{x_o | x_e} x = \min_{x_e} \{(1 - c) \cdot x_e + c \cdot \hat{x}_e\}$$

Since Theorem 2 asserts $x_e \ge \hat{x}_o$, we have

$$\min_{x_e} \max_{x_o | x_e} x = (1 - c) \cdot \hat{x}_o + c \cdot \hat{x}_e = \hat{x} \tag{5}$$

When the edge chooses $x_e = \hat{x}_o$ (also achievable in reality; see §5.2 for the details). Similarly, we can prove that, the operator's maximin strategy will also result in $\max_{x_o} \min_{x_e | x_o} x = (1-c) \cdot \hat{x}_o + c \cdot \hat{x}_e = \hat{x}$ if $x_o = \hat{x}_e$, that is

$$\min_{x_e} \max_{x_o | x_e} x = \max_{x_o} \min_{x_e | x_o} x = \hat{x} \tag{6}$$

i.e., both the operator and the edge's best local strategy will result in $x = \hat{x}$. This decision ($x_e = \hat{x}_o$, $x_o = \hat{x}_e$) is also the *only pure Nash Equilibrium* for Algorithm 1: If the edge deviates $x_e = \hat{x}_o$, the above procedure ensures that it result in more than $\hat{x}$ charging. So if both the edge and operator are rational, Algorithm 1 will return with $x = \hat{x}$. □

D TLC IN THE GENERIC DATA CHARGING

Algorithm 1 applies to generic uplink network traffic. For the downlink traffic, it applies if the server is co-located with the cellular core (§4). This holds for the cellular edge servers, but not in the generic setting. Otherwise, downlink data loss may occur between the edge server and 4G/5G core. Over-charging may still occur, but Algorithm 1 ensures that it is still *bounded by* the loss between edge server and core.

Specifically, let $\hat{x}'_e$ and $\hat{x}_e$ be the real downlink data volume at the Internet server and 4G/5G core. So we have $\hat{x}'_e \ge \hat{x}_e$ in general. Ideally, the edge should be charged based on core-received data $\hat{x}_e$ (thus $\hat{x}$ due to Theorem 3). But the edge's report will result in $\hat{x}' = \hat{x}_o + c \cdot (\hat{x}'_e - \hat{x}_o)$. So the edge will be over-charged by

$$\hat{x}' - \hat{x} = c \cdot (\hat{x}'_e - \hat{x}_e)$$

i.e., the edge will be over-charged *no more than* the data loss between edge server and 4G/5G core. A selfish operator *cannot* over-charge more than that. This result is still better than the legacy 4G/5G, which allows unbounded over-charging.