Edge Computing in the ePC — A Reality Check

Ilija Hadžić
Nokia Bell Labs
600 Mountain Avenue
Murray Hill, NJ 07974
ilija.hadzic@nokia-bell-labs.com

Yoshihisa Abe
Nokia Bell Labs
600 Mountain Avenue
Murray Hill, NJ 07974
yoshihisa.abe@nokia-bell-labs.com

Hans C. Woithe
Nokia Bell Labs
600 Mountain Avenue
Murray Hill, NJ 07974
hans.woithe@nokia-bell-labs.com

ABSTRACT

Mobile Edge Computing (MEC) has received much attention from the research community in recent years. A significant part of the published work has studied the telecom-centric MEC architecture, which assumes that the computing resource is located at the edge of the mobile access network (e.g., the Evolved Packet Core), typically at the first aggregation level. Many authors make a silent assumption in their analyses that the latency at this stage of the network is negligible. In this paper we show not only that this assumption false, but that in some common cases the latency of the first-aggregation stage dominates the end-to-end latency. We challenge the latency argument in the context of present-day access networks and discuss what must be done to pave the way for practical deployments of MEC.

CCS CONCEPTS

• **Networks → Wireless access points, base stations and infrastructure**; **Network measurement**;

KEYWORDS

LTE, MEC, latency, edge computing, cloud

ACM Reference format:
Ilija Hadžić, Yoshihisa Abe, and Hans C. Woithe. 2017. Edge Computing in the ePC — A Reality Check. In *Proceedings of SEC '17, San Jose / Silicon Valley, CA, USA, October 12–14, 2017,* 10 pages.
DOI: 10.1145/3132211.3134449

1 INTRODUCTION

After a decade of centralizing computing resources in what is today commonly referred to as "the cloud," the technical community is showing signs of trend reversal. One such reversal is that of edge computing[11], which calls for installing shared computing resources (cloudlets) at the edge of the network, in the proximity of the user. The primary motivation in this case is to bridge the performance gap between mobile and stationary computing resources, through delegation of computation from the former to the latter, while supporting interactive applications[17].

Computing at the edge of the network has two main advantages. First, it confines bandwidth demands in cases where geographic locality of user devices exists. In other words, what caching does for stored content, edge computing does for real-time generated content. Second, it reduces the round-trip latency between the end-user device and the cloud-resident resource, and thereby enables the offload to the cloudlet of latency-sensitive applications that are not offloadable to resources deeper in the cloud. Although the computing power of hand-held devices is growing, application offload is still desirable because the demand of applications is also growing, and the gap between mobile and stationary resources is not narrowing[16].

Edge computing as a general concept does not impose restrictions on the specific location of the cloudlets other than that they should be "at the edge of the network" or "in the user's proximity." Who owns and manages the cloudlet and on whose premises it physically resides is an ongoing debate; different authors have made different assumptions in their work. While this detail may seem minor, it actually has broad technical and operational implications. A cloudlet that resides inside the access network, owned and managed by the service provider, is very different from a computing node that resides in the user's home and is willing to offer its computing resources to any mobile user in its vicinity.

In this paper we consider the telecom-centric MEC architecture[15], which offers new business opportunities for the telecom providers. The edge, in this case, is located at the first-level of aggregation in the access network. In a mobile network, that location would ideally be a base station, although practical limitations of outdoor installations often push the server pools deeper into the backhaul network, also known as the "evolved packet core" (ePC) in LTE terminology. Much of the published work presumes negligible latency between the user and the compute node. We performed a series of measurements that challenge the validity of this assumption. We find that a naive design approach can easily result in deployments in which the access latency is dominant, and we discuss what can be done in current and future mobile networks to address the latency problem.

2 RELATED WORK

The telecom-centric MEC architecture is advocated and standardized by an industry consortium sponsored by ETSI[15]. Many authors in the research community have adopted this model, with similar assumptions about latency at the edge.

In [19] the authors present a high-level overview of MEC architecture that presumes the telecom-centric model. They also make a tacit assumption that the latency associated with the radio link is negligible or nonexistent.

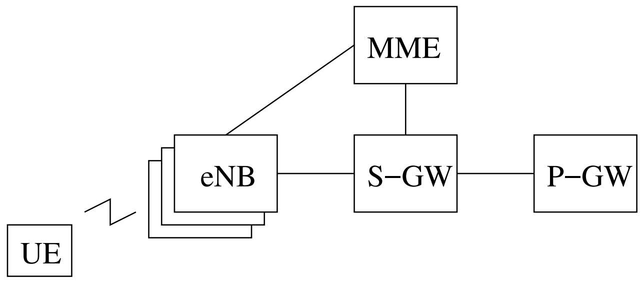

Figure 1: LTE backhaul (ePC) architecture.

Chang et al.[5] implemented a cache system using an ETSI-compliant MEC platform and an open-source implementation of a base station[14]. Despite non-negligible latency of the LTE link, improvements were significant because the remote data center was on the other continent. In practice, it is unlikely that the penalty of fetching content from a remote data center (e.g. cloud) will be that high, because major cloud providers operate multiple data centers around the world.

Mao et al.[12] derived an energy consumption model for user devices under the assumption that the application is augmented by a MEC server located inside the mobile access network. Their detailed model factors in the wireless link bandwidth limitations and channel impairments, but it assumes that the execution time at the MEC server and the network latency to that server are negligible.

The GENI project[4] is an experimental distributed cloud testbed that includes edge-computing capabilities. The system provides user-accessible compute nodes along with private LTE and WiMAX networks on which researchers can experiment. The infrastructure has been used to demonstrate the application of MEC to control connected autonomous vehicles[8]. It is interesting that this work is one of the rare cases in which the authors acknowledged that latency is a significant problem and that the latency often comes from the first access node.

Nguyen, et al.[13] studied the TCP performance in cellular networks in simulation environment. The study assumed 3 ms one-way delay for the wireless access link, which is one half of the theoretical minimum round-trip delay. In practice, the delay this short is never achieved.

The survey presented in this section is not exhaustive, but it provides strong evidence that interest in telecom-centric MEC architecture is significant. With latency being a critical performance metric, we must ensure that we do not naively assume that simply putting the server at the edge of the access network automatically solves the latency problem simply because the client is only "one hop away."

3 SERVER LOCATION

Fig. 1 shows a simplified view of a modern mobile access network (LTE in this case).

Base stations, also referred to as "eNodeB" (eNB), form clusters within which they coordinate handoffs without the involvement of any element at a higher hierarchical level. Once the serving cell is determined for a given user (UE — short for "user equipment" — in the figure), a GTP tunnel is established to the serving gateway (S-GW). This tunnel moves as the UE is handed off from one eNB to another. The S-GW connects the tunnel to another that it establishes to the packet gateway (P-GW). This tunnel remains intact as long as the UE remains in the same cluster (and hence maintains the same S-GW). If the UE changes cluster, the S-GW also changes, which requires both tunnels to change, but the selected P-GW remains the same. The handoffs across clusters require coordination on the higher hierarchical levels. This coordination is handled by the mobility management entity (MME).

Although the ePC consists of many IP routing hops, none of them is visible from the public internet. The UE gets either a private-subnet IP address that is translated to a public address at the P-GW (typical for IPv4-based networks where addresses are scarce), or a public IP address that is tunneled through the ePC to emerge out of the P-GW (typical for IPv6-based networks where addresses are abundant). In either case, an Internet-based (over-the-top) application perceives the entire ePC as a single routing hop.

Adding commodity servers to the ePC has design and operational implications. Unconstrained latency optimization may lead to the naive conclusion that each eNB should have a dedicated and physically co-located server farm. Doing so is often impractical, because base stations are typically installed at inaccessible locations with limited physical space for equipment. A macro-cell tower may be on the top of a hill or near a highway with limited real-estate available for equipment. Metro-cells are almost always installed in open space (lamp posts, walls, street cabinets, etc.) without any rack space for a commodity server or any kind of IT-grade infrastructure. Supporting the co-location of the eNB and MEC server would likely require a redesign of base-station sites to accommodate the server equipment. While theoretically possible, the effort would be significant and would have to be well justified.

Alternatively, the MEC server can be located on the other end of the fiber at the central-office (CO) site, but still dedicated to the base station or a small group of base stations. In either case, the each server farm would have to be economically justified by sufficient traffic aggregation. Moving servers deeper into the ePC results in higher traffic aggregation but increases latency. From the utilization perspective alone, the best place for the server farm is a single centralized location. But, if the location is "too centralized," then latency can be too high.

To complicate things further, placing the server farm anywhere inside the ePC requires either interacting with the ePC signaling system and tapping into the GTP tunnels, or building a platform that integrates the S- or P-GW functionality with the application runtime[9]. Compared to an over-the-top solution, placing the MEC farm in the ePC adds complexity and requires addressing a number of practical problems, mostly related to the physical design of the system. In an over-the-top solution, the natural place for the server farm is behind the P-GW, using public IP connectivity, and a MEC node is accessed just like any data center on the Internet.

An over-the-top solution is, by the nature of the design, more centralized than any deployment inside the ePC, which plays well with the bandwidth-aggregation and server-utilization argument. On the other hand, a deployment inside the ePC plays better with the bandwidth-confinement argument. If there is a small group of cells generating high volume of computing demands, a local server acts as a generalized cache node, providing either local content or local computing power.

The remaining question is that of latency: Is the potential latency reduction worth the effort? Our objective is to answer whether a MEC node inside the ePC, preferably at the base-station site, delivers sufficient latency gains to justify the tight interaction between or complete integration of the compute and transport infrastructures.

We perform a series of delay measurements and analyze the results to understand the contribution of each ePC component. Rather than using an artificially constructed lab setup, which is arguably always trivial compared to an actual network deployment, we perform our measurements on a commercially deployed network and analyze the data to determine the components that contribute to the delay. Probing the "real world" network yields the experimental results that reflect not only the limitations of the communication protocol (LTE standard in our case) and equipment, but also the network parameters that a typical network operator may use to address various technical and economic constraints.

From the analysis we can determine how much end-to-end latency can be eliminated by moving the server farm closer to the user. These results allow us to quantify the potential advantages of the telecom-centric architecture compared to an over-the-top system.

4 DELAY MEASUREMENTS

The simplest and most straightforward way to determine the delay through an ePC network, without access to any information about the internal design of the network, is to measure the round-trip times (RTT) from the UE to the first hop behind the P-GW, using the common `ping` utility.

To try to identify the IP address of the target hop, we ran the `traceroute` from the UE to an arbitrary IP address on the public Internet. The ePC is not visible to `traceroute` because the routers inside the ePC are hidden by the GTP tunnels. The first few hops that are visible are typically IP addresses owned by the carrier and the delay differences between these hops reported by the `traceroute` are typically small (within a millisecond). We assume that these nodes are a good representative of the closest location where an over-the-top provider can install a server farm. Any installation closer to the UE would have to be inside the ePC and intercept the GTP tunnels.

We would ideally like to ping the first visible node in the path, but sometimes that node did not respond to ICMP echo requests; in such cases we pinged the next node in the path. Because of the presumed proximity of the two nodes, the results were not significantly affected. Fig. 2 shows the observed delays over a one-hour period.

The delays are not only high and highly variable, but they also follow a periodic sawtooth pattern. This systematic beating typically points to the presence of slotted links in the communication path and has been well documented in the synchronization and timing technical community[10]. By "slotted" we mean a link that transmits at periodic intervals. A typical example is a legacy TDM or SONET/SDH link, but many modern links, such as EPON/GPON, DOCSIS, and wireless access links, exhibit the effects of slotting.

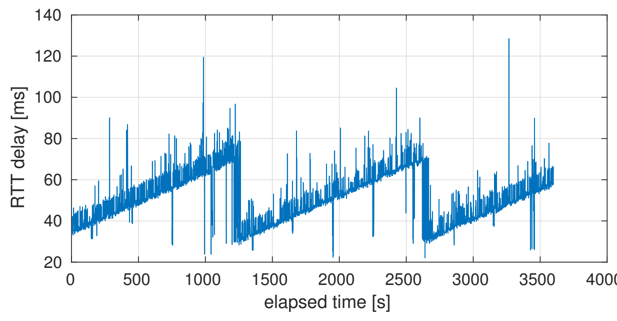

Figure 2: Observed RTT times using common ping utility.

A sawtooth delay pattern forms when both the packet-generation process (the `ping` utility in our case) and the transmission link exhibit periodic behavior, but the clocks of the two processes are not synchronized. The packet-generation grid then slides in time relative to the link-slot grid. Packets generated immediately before a transmission slot are sent right away, whereas packets generated immediately after a slot exhibit a delay equal to the slot period.

The delay pattern of Fig. 2 suggests the presence of a slotted link, but it is also possible to deduce the contribution of other network components. To do that we repeat the ping test in a a setup carefully designed to control the timing of packet generation.

4.1 Identifying the Slotted Link

Slotted links can exist anywhere in the access network, but we hypothesize that the likely source is the LTE link itself and design the test to prove (or disprove) this assumption.

LTE base stations are typically synchronized to a UTC-traceable clock (often using a GPS source)[1]. Hence, the slotted-link effects will only be visible if we can control the timing of packet generation using a GPS clock. We synchronize the UE host (a PC running Ubuntu 16.04 Linux, with an LTE modem connected to its USB interface) to an NTP server, connected over a direct Ethernet wire, whose clock is derived directly from a GPS receiver[18]. We also modified the `ping` utility (version 3:20121221-5ubuntu2) to send packets at precise time intervals — behavior not guaranteed by its original code. First, we changed the granularity at which the program keeps track of time, from milliseconds to microseconds. Second, we changed the code to busy-loop, checking whether to send the next ping packet and whether a reply packet has arrived. (Originally, it can sleep inside this loop of sending and receiving packets; although this behavior is fine under usual circumstances, in our case it made the timings of packet generation deviate too much from the intended values.) With this setup we can produce a sequence of ICMP packets whose generation grid is precisely controlled relative to the slots produced by the LTE link.

The blue line of Fig. 3 shows the measured round-trip latency using ICMP packets sent every 39.9 ms from the UE to the first visible node behind the P-GW. We selected this packet-generation period expecting to provoke a sawtooth pattern that we can match with theoretical predictions. Namely, when packets are sent over a

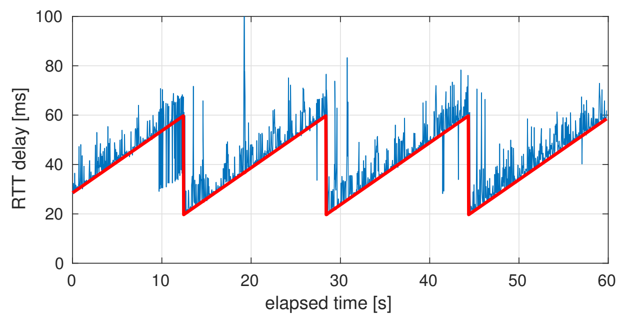

Figure 3: RTTs using precise ping and synchronized UE, T_p = 39.9 ms.

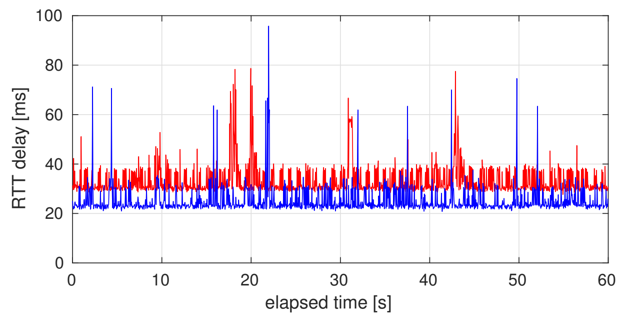

Figure 4: RTTs using precise ping and synchronized UE, T_p = 40.0 ms.

slotted link every T_p units of time, the delay of pth packet is given by

$$t_d[p] = \left(\left\lceil p\frac{k}{n} - \varphi\right\rceil \frac{n}{k} + \varphi\frac{n}{k} - p\right)T_p, \quad (1)$$

where k and n are coprime positive integers such that the time between slots is given by $T_s = \frac{n}{k}T_p$, and $\varphi \in [0, 1]$ is the relative phase shift expressed as the fraction of T_s, between the packet-generation grid and the link-slot grid. The formal analysis and derivation of (1) is provided in Appendix A. As discussed there, the observed sawtooth amplitude (40.0 ms in our case) equals the slot period T_s. Thus, setting T_p = 39.9 ms is expected to produce delays that match the predictions of (1) for k = 399 and n = 400.

The red line of Fig. 3 shows this prediction for $\varphi = 0.22$.[1] The parameter φ must be fitted by trial and error because its value simply reflects the time when the experiment was started relative to the position of the slots. Next, we set the packet-generation period to exactly 40 ms (the suspected slot period) and the sawtooth has disappeared. As expected (see Appendix A), this experiment produced a delay bias that changes with each measurement session. The blue and red line of Fig. 4 show two such experiments.

[1]The red line is a plot of (1) for the given values, but to which a delay bias (i.e., y offset) has been added to match the bias of the blue graph. The source of the bias is explained in Appendix A.

This set of measurements is sufficient evidence that somewhere in the path between the UE and the first visible node after the P-GW there exists a slotted link with a 40 ms slot period. To understand the delays introduced by the ePC, we need to understand whether the slotting is inherent to the UE, eNB, or some other component of the ePC or the remainder of the path.

4.2 Cause Analysis

As previously stated, our hypothesis is that the dominant source of sawtooth delay pattern is the LTE access link. The rationale is that repeated tests over wired DOCSIS (cable) and GPON (fiber) networks did not exhibit this delay pattern, and legacy TDM links (if the ePC had any) have a slot period of 125 μs rather than the observed 40 ms.

The packet rate we use is sufficiently high to force the UE to remain in active and synchronized state during the test. On the other hand, the rate is sufficiently low that it does not provoke packet bunching on non-slotted (e.g., Ethernet) links or rate-throttling in the operator's network. We can therefore rule out any random-access-channel periodicity, as well as any delays associated with connection setup or state transitions.

The next likely cause to examine is the uplink scheduling protocol. Because we are using a data-only best-effort service, the network will likely place our traffic on the default bearer channel with dynamic scheduling. For uplink transmission to happen, the UE must first place a scheduling request (bid) onto the uplink control channel, after which it will receive a grant that specifies a bearer transmission opportunity.

The bid-to-transmission delay depends on the available uplink resources, background traffic, and scheduler policy. On an uncongested network with a typical scheduler, this delay should almost always be close to the minimum of 5 ms. Although the scheduler is allowed to do whatever it deems necessary to get the traffic through, including assigning a periodic schedule, the pattern produced by the scheduler would be a function of the cell traffic — unless the scheduler is traffic-oblivious (e.g., strict round-robin), which we consider highly unlikely. Repeating the test at different times of the day including the night and weekends always produced the same general sawtooth pattern, which leads us to conclude that the observed sawtooth delays are oblivious to the traffic from other devices in the cell and therefore not caused by the scheduling policy.

After reasoning out other potential causes related to the LTE link behavior, we hypothesize that the primary source of the sawtooth behavior is the periodicity of the scheduling-request channel (SR periodicity). Namely, the uplink control channel allows placing bids in specified time slots that repeat with a periodicity established at connection setup time. The periodicity can be as low as 1 ms and as high as 80 ms, where lower values allow for lower latency at the expense of supporting fewer active UEs per cell[3].

To verify our hypothesis we ran measurements in lab conditions using an isolated base station on which we have full control of the cell parameters, including the SR periodicity. The entire ePC was emulated in software on one Linux server; the delays through the server were (measurably) negligible. In this lab setup we can fully

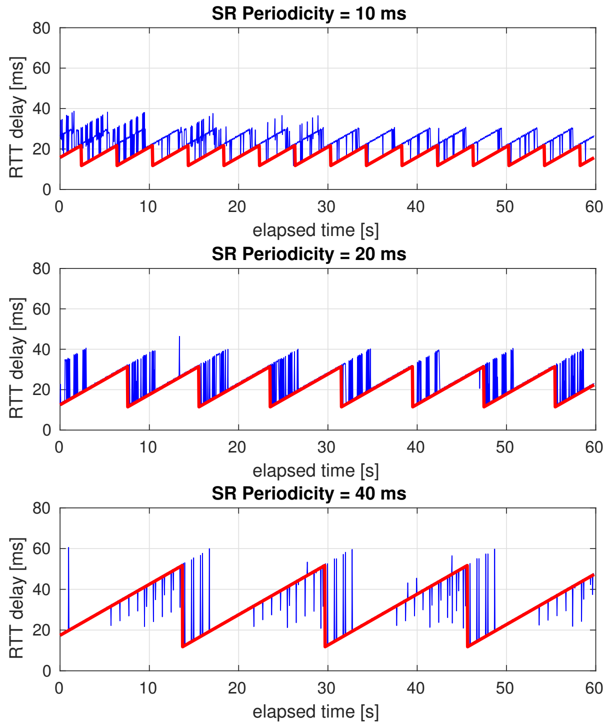

Figure 5: RTTs in lab conditions with controlled SR periodicity.

attribute the delays measured by pinging the P-GW to the components of the air interface. These components include the UE device driver, the UE modem, the eNB modem, and the packet-processing stage of the eNB. Fig. 5 shows the measured delays for different values of SR-period configured at the base station.

As in the earlier figures, the blue and red lines respectively represent measured data and theoretical predictions. For each test we used $T_\mathrm{p} = 39.9$ ms, so we had $k = 399$ and $n \in \{100, 200, 400\}$. The almost-perfect match of the delay is strong evidence that SR periodicity is the dominant source of the observed delay pattern.

Each discontinuity is accompanied by a set of ripples, which we attribute to the delays through the UE host. Namely, when the `ping` utility submits the packet to the socket, the packet must propagate through the software stack (typically involving the USB driver), USB bus, and the modem card itself. This delay is random and depends on other activity in the operating system.

To demonstrate that the software uncertainty is indeed the source of the ripples we extended the delay model (see Appendix A for details) to the following,

$$t_\mathrm{d}[p] = \left(\left\lceil p\frac{k}{n} + \delta - \varphi \right\rceil \frac{n}{k} + \varphi\frac{n}{k} - p\right)T_\mathrm{p}, \tag{2}$$

where the new dimensionless parameter δ is a random variable that represents the ratio between the software-stack delay and the slot period T_s. Determining the exact random process that generates δ is a difficult task and outside the scope of this paper, but we were able to closely reproduce most of the ripple patterns shown in Fig. 5 by making δ a linear combination of multiple Bernoulli and uniform random variables.

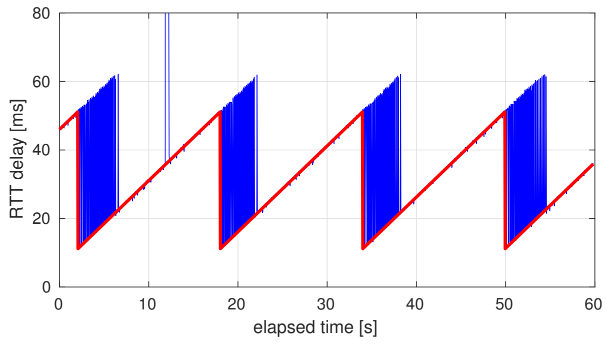

Figure 6: Repeated RTT measurements for 40ms SR periodicity on a different base station.

The general appearance of the delay plot for 10 ms periodicity looks different from the plot for the other two cases; this difference is the result of δ being comparable to T_s. One thing that δ does not explain well is the the presence of delay dips that happen before the discontinuity in the bottom plot (40 ms periodicity). These dips are in contrast to the ripples that occur following the discontinuity, which are well explained by the model. We believe that these dips result from the internal operation of the scheduler, which (we believe) simply decides to grant the slot sooner relative to the location of the nominal SR-channel slot. Repeating the same test with a different base station design model (Fig. 6) showed no dips for this test case.

5 DELAY BREAKDOWN

We established thus far that the primary source of the delay and its variability is the SR-channel periodicity. We further investigate how it contributes to the final latency experienced by the UE.

5.1 Piggyback Effect

Devices that generate small packets relatively infrequently, as is the case for many sensors and other types of small devices, are especially vulnerable. In such cases the upstream scheduler at the base station will typically not issue the device any unsolicited grants, and hence the device will have to wait for up to an entire SR period to place a bid, after which it must then wait for an unspecified amount of time to receive the desired grant.

Sometimes a low-rate source at the UE can benefit from the background traffic produced by another process hosted at the same UE. One likely manifestation of this effect can be seen in Fig. 3 between time 10 s and 12 s. The prolonged burst of lower delays would occur because the testing host was a general purpose Linux machine and another background application initiated its own network traffic that caused a higher bid to be placed on the SR channel. This phenomenon would cause the scheduler to assign multiple upstream grants that were spread between the current and next SR-channel slots. Because the UE modem makes no distinction between applications mapped to the same bearer channel, one of the earlier grants was used by the Internet Control Message Protocol (ICMP) packet, resulting in an earlier transmission.

To substantiate this claim we produced the data of Fig. 5 with an `iptables` rule that blocked on the LTE network interface all outgoing packets except ICMP. The middle trace (20 ms periodicity) was completely dip-free for the full duration of the test and for all repeated tests (the full dataset that we collected was much longer than the 60 seconds shown in the figure), whereas the dips in the 40 ms trace are too regular to be coming from random traffic. They are more likely to come from the internal operation of the scheduler, which is allowed to issue any grants at any time that it likes as long it delivers the expected service-level agreement.

5.2 Upstream versus Downstream

We now seek to determine how much of the delay is caused by the upstream versus the downstream path. We suspected, because of SR periodicity, that the upstream is the primary contributor. To confirm this hypothesis, we performed one-way measurements using a dual-homed Linux container hosted on the same physical node as the ePC in our isolated environment.

We reworked our test setup such that the UE and ePC are hosted on the same physical machine but in different containers. The container hosting the UE had two network interfaces. The first interface was virtual and was bridged off the same physical interface as the P-GW (which runs in a separate Linux network namespace). This setup gave us a "backdoor" path between the UE and the P-GW with negligible delays. The second interface of the UE container was the physical LTE device. We configured the `rp_filter` and `accept_local` kernel parameters to prevent ICMP packets from being dropped in the dual-homed environment.

In the experiment an ICMP echo request leaves the UE container via the LTE interface and reaches the P-GW, which relays the request back to the container via the backdoor path. The UE container sends an echo reply to the P-GW, also via the backdoor path, and the P-GW relays the reply back to the LTE interface.

We used `tshark` to capture the ICMP packets on both container interfaces. With this machinery we can perform the measurements as before, but we are able to separate the downstream and upstream traffic by analyzing the packet-capture files. The minimum downstream delay that we observed was 4.8 ms and the median was approximately 5.7 ms. Of the observed 4.8 ms, at least 1 ms is fundamental to the protocol (the minimum delay introduced by the downstream scheduler). There was no sawtooth pattern in the downstream delay, whereas the delay in the upstream path was very similar to the RTT figures presented earlier. These results confirm the expectation that SR periodicity is the dominant contributor to round-trip delay.

5.3 Additional Delay Sources

To understand other delay contributors, we examine the delay histogram for traces of Fig. 3 and Fig. 4. In the former case we first remove the sawtooth by subtracting the envelope (i.e., the red line of Fig. 3). We also throw away the dips as outliers. The resulting signal is the delay trace that would happen if the influence of SR periodicity were not present. For all measurements collected, the distribution was very similar and had the shape shown in Fig. 7.

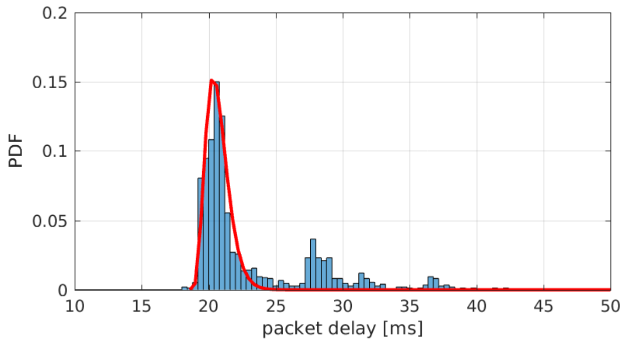

Figure 7: Round-trip delay distribution.

The distribution is trimodal with the modes spaced apart by approximately 8 ms. The modes arise from link-level retransmissions. By protocol, if the transmission on the LTE link fails, a negative acknowledgment (NACK) is sent 4 subframes later (one subframe is 1 ms long) and (HARQ) retransmission occurs 4 subframes after receiving the NACK[2]. The modes of the distribution represent the number of retransmissions it took for a packet to get through the LTE link.

The histogram floor (which is the delay bias) represents the aggregation of all constant delays, including the propagation time over the air, wires through the ePC, and packet-processing circuits of the base station, UE, and routers that constitute the ePC. From Fig. 7 that delay was 16.5 ms in our measurements. It is interesting to compare the minimum delay observed in the live network with the minimum delay observed in the lab. We read the latter from Fig. 5 and get approximately 11.5 ms for all three SR-periodicity cases. If we attribute the 5 ms difference to wires and fibers in the ePC, we arrive at a rough estimate of the geographical span of the ePC; it is approximately 500 km, assuming the speed of light through the fiber of 2×10^8m/s. The geographic span is actually somewhat smaller, because the estimated delay includes internal wiring in the switching centers and internal propagation through the packet processing circuits (excluding queuing) that implement the equipment.

If we focus on the first mode, we get the distribution of packet queuing delays through the ePC. In this segment of the network we expect to see more traditional packet switches and routers, and so a simple but realistic model of the delay is a sum of exponentially distributed queuing delays for each network hop. Hence the expected delay distribution through the ePC should resemble an Erlang distribution. In practice deviations will occur because the delays at each switch are neither exactly exponentially distributed nor identical; however, fitting an Erlang distribution over the first mode of the histogram should give a good idea of how much the ePC without the LTE link contributes to the delay.

The red line in Fig. 7 shows the Erlang-6 distribution for $1/\lambda =$ 0.35ms with an added 18.6 ms bias fitted over the first mode of the measured delay distribution. We performed the fitting by generating the samples of Erlang PDF function at intervals equal to the histogram bin spacing, followed by multiplying each sample value by the bin width. The resulting product is the probability mass that is comparable to the histogram bar heights of the measured data. We further multiplied the PDF values by the sum of the histogram bars of the first mode. The reason we had to do this additional scaling of the red line is because the histogram is normalized to represent the probability of the delay falling into the corresponding bin. Because the bars add up to 1 and the Erlang PDF also integrates to 1, scaling is necessary to make the red line fit the first mode of the histogram PDF. This fit gives a good sense of the queuing and propagation delays in the ePC; the floor of the fitted distribution is at 18.6 ms and the 99th percentile is at 27 ms, resulting in a queuing delay of about 8–10 ms.

The round-trip delay through a commercial LTE network is significant and can be as high as 80 ms, depending on the configuration. For our commercial LTE network, only 15 ms can be attributed to the propagation and queuing in the ePC (5 ms for wires and fibers plus 10 ms for queueing). Most of the delay comes from the uplink, with SR periodicity being the main culprit, at least for low-bandwidth traffic. In our experiments on the live network that delay was 40 ms. Each HARQ retransmission adds 8 ms of penalty in the upstream. This set of delays can be, to some extent, controlled by tuning the network parameters.

From experiments in the controlled environment, we found that there is approximately 11 to 12 ms of irreducible delay, some of which can be attributed to the protocol. In the upstream there is a minimum of 5 ms scheduling delay, which is the time it takes for the UE to receive a grant after transmitting a bid[2]. In the downstream the minimum delay is 1 ms. The scheduler can introduce additional delay by delaying its issuing of grants. Subtracting $12 - (5 + 1)$ leaves no more than 6 ms (roundtrip) to attribute to the eNB and UE implementation details, which is not a significant value compared to all other delays.

The key takeaway is that, in the most optimistic case, the latency of the first hop (UE and eNB combined) is of a similar order of magnitude to that of the latency through the ePC. In a more realistic scenario the first hop is the primary source of latency. It is also interesting to evaluate how the observed latency in the ePC compares to the Internet latency to a public data center. We compared our ePC ping results with the results of pings to an Amazon AWS instance in their East Coast data center (the closest location from where we did the experiments), and we found that the Internet (in this case) adds approximately 10 ms to the overall round-trip delay — a similar order of magnitude to that observed inside the ePC.

6 DELAY SIGNIFICANCE

The delay pattern of Fig. 2 is typical when the communication path contains a slotted link and the packet-generation period at the application level is not an integer multiple of the slot period. In practice this phenomena happens when the application host clock is not synchronized with the network clock. In our measurements the host is synchronized so that we can precisely control the timing relationship between packets and slots, but the practical consequence is for unsynchronized applications, which constitute the vast majority.

Suppose that we have a sensor or a human-input device that decides to transmit an event, but that has no knowledge of the network clock nor of the temporal location of the transmission opportunities — a typical case for applications driven by the available sensor data (IoT) or human actions (interactive applications). The event will reach the server with a delay ambiguity that equals the slot-periodicity value on the top of all the other delays associated with the LTE uplink protocol. After they leave the UE, the packets will be paced by the temporal location of the slots, rather than by the data source itself[6].

The delay through an LTE network is often associated with transitioning the UE from the idle (i.e., low power) state to the active state, or with the slotted nature of the link when the UE is actively transmitting. The former case occurs when the UE has been idle for a prolonged period (typically several seconds); the delay to return to the active state ranges from tens to a hundred milliseconds. The delays in the latter case are on a much smaller scale and are determined by the subframe period: If the UE always has data to send and the scheduler is willing to grant slots to it, the transmission can occur on any subframe boundary (1 ms).

The significance of the example that we demonstrated is that there is a whole class of applications that keep the UE in the active state but still experience first-hop delays that are dominant compared to the total latency through the ePC. Consequently, placing the server at the base station site will still leave a large portion of the latency problem unsolved.

The vulnerable applications are those that periodically send low-volume data and whose period ranges from a few tens of milliseconds to a few seconds. Unfortunately, many applications of interest for MEC fall into this category. Typical cases include applications that interact with humans at video frame rate. Take, for example, a VR headset that receives a viewport video from the server and sends updates about the position of the user wearing the headset. Typically the uplink data is low volume and is sent at the rate that matches the video frame rate. This traffic pattern falls exactly into the category that we describe as vulnerable.

There are several mechanisms that can be used to mitigate the problem. The simplest and the obvious one would be to reduce the SR-periodicity value. The standard[3] allows this value to be as low as 1 ms — at the expense of a reduced number of users per cell. Using a semi-persistent schedule would help hide the effects of the SR channel, but the delays would be still as large as the schedule period, which must be at least 10 ms. In either case, coordination between the UE and base-station scheduler would in general be necessary to further hide the slotted link delay.

Fabini[7] proposed a method for hiding slotted link delays, where the application measures the delays, locates the sawtooth minimums, and samples the data at moments that immediately precede the minimum-delay point. This method requires no coordination with the network, but it is limited to applications whose data-generation period is an integer multiple of the slot period.

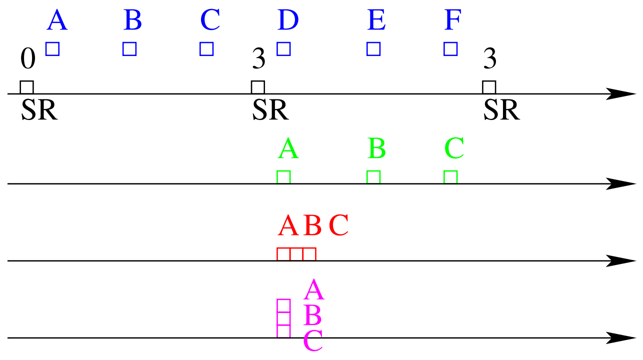

Figure 8: The impact of the upstream scheduler on latency.

In general, to eliminate the SR-channel-induced delays, the scheduler must be coerced into assigning uplink transmission opportunities that "coincide with" the UE packet-generation events. This property may be difficult or even effectively impossible to achieve, depending on the nature of the packet-generation process and the design of the scheduler. Consider, for example, a case in which an application generates periodic events, represented by the blue letters shown in Fig. 8. Suppose for simplicity that each event exactly fits one transmission opportunity. As these events queue up, the modem places into the next available SR slot a bid for three opportunities.

The scheduler has three options (among others), shown in different colors in Fig. 8. It can evenly spread the assigned opportunities, or it can bunch the opportunities, either in time or across subcarriers. In the first case, the delays will be constant and equal to the SR period. In the other two cases, the delays will follow the biased sawtooth pattern (second special case in Appendix A). None of these cases is desireable for MEC applications.

What is necessary is the ability for the application to notify the UE modem of how many events it intends to generate in the next SR period, for the modem to bid this request ahead of time, and for the scheduler to evenly spread the assigned opportunities. In Fig. 8 the bid would be placed in one SR slot earlier than shown. Enhancing the application is possible for those that periodically generate sensor samples (e.g., the majority of IoT applications and interactive VR applications), but not for those that must handle truly asynchronous events (e.g., a human pressing a button). The scheme also relies on a particular upstream scheduler policy, over which the UE has no control. It also requires a modem redesign — commercially available modems do not support any coordination with upper application layers.

Another alternative is to generate, on the UE, a low bandwidth constant-bit-rate flow of packets that will ensure that the modem will always have enough packets in the queue to cause it to place a non-trivial bid in every SR slot. This packet flow is not used for anything except ensuring that there are enough opportunities assigned to the UE. At the socket level, the UE can give higher priority to the event flow, so that whenever an event-related packet is placed on the socket, it is delivered to the modem first. In this way, the events "ride the ticket" of the other flow. We tried this experiment in the lab and successfully reduced the ping times to match the observed floor delay. The downside of this method is the power and bandwidth penalty: The UE is now transmitting data that carries no useful information.

7 CONCLUSIONS

Although our measurements were not exhaustive across locations and service providers, we believe that they paint a realistic picture of the readiness of commercial LTE deployments for edge-computing. Our findings challenge a somewhat widespread belief that simply placing a server at the base station site solves the latency problem.

Based on our experiments, the first hop (UE to eNB) introduces an irreducible latency of roughly 10-12 ms and adds a sawtooth pattern with an amplitude of about 40 ms. Further, each HARQ retransmission adds another 8 ms of latency. The ePC segment adds about 15 ms, and if the server is located in a public data center, the Internet segment adds theoretically unbounded delay, but without much effort we were able to reach a data center that was within 10 ms round-trip delay. It is reasonable to assume that a latency-sensitive over-the-top solution would use a network of distributed data centers that would keep the Internet delays at the level that is comparable, if not lower, than what we observed in the access network.

Common sense tells us that the first problem to address is the air interface. As we discussed, some of the observed delays can be removed by changing network parameters or carefully redesigning the scheduler policy and how the UE places upstream bids for bandwidth. Both issues are addressable without changing the LTE protocol. However, what is theoretically achievable is not necessarily practical in the field. Network operators design and run their networks constrained by the economic factors more than the technical ones. The SR periodicity, which we discussed at length, is an example. The standard allows this parameter to be as low as 1ms, which would have direct positive impact on roundtrip delays. However, the latency reduction would come at the expense of severely limiting the maximum number of users per cell. Thus, the operators are not in position to use the lowest allowable value for this parameter and, consequently, the equipment vendors rarely implement base stations that support this value. Similar limitations will hold when 5G technologies that promise sub-millisecond latency are deployed. It is not sufficient to simply define a low-latency mode by the standard. The implementation must be practical in a commercial network.

After the first-hop problem is addressed, moving the server farm closer to the P-GW and, thus, eliminating the Internet segment of latency is the next logical step to take before actually entering the ePC. Within the ePC we have an opportunity to remove another 15 ms of latency, but at the expense of increased complexity through having to interact with the ePC control plane and intercept the GTP tunnels. The gain of this 15 ms must be significant compared to all the other delays to make it worth attacking.

8 ACKNOWLEDGMENTS

Many individuals provided invaluable input that helped us improve the content of this paper. We are grateful to Marcos Tavares, Dragan Samardžija, Harish Viswanathan, and Krishan Sabnani (*ret.*) of

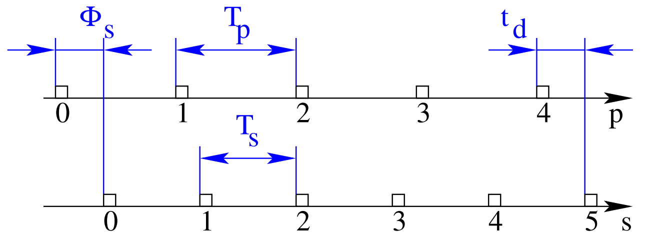

Figure 9: Packet-generation and link-slot timing relationship.

Bell Labs. We are especially grateful to Martin Carroll, also of Bell Labs, for in-depth review and for volunteering his time to provide numerous edits and improvements to the paper.

A SLOTTED LINK ANALYSIS

Packets sent periodically (e.g. ICMP echo requests sent using the `ping` application) over a slotted link incur delays given by (1). Here we derive and analyze this expression. For sporadic short packets sent over an LTE link, the effects of a slotted link occur in the uplink direction and are caused by the SR-channel periodicity as discussed in Section 4. However, the analysis provided here is general and not restricted to the LTE channel.

Let T_p be the period between the generated packets intended for transmission and let T_s be the period at which the transmission opportunities repeat on the slotted link. Also, let Φ_s be the initial phase shift between the packet-generation grid and the link-slot grid — specifically, when the first packet of a test is generated, the first available transmission slot is at time $t = \Phi_s$. We also introduce a packet counter variable p and a slot counter variable s, with both counters starting at 0. The relationship among these variables is illustrated in Fig. 9.

We assume that when the packet is generated, it is transmitted in the next available slot, and that there are no slot-size restrictions or queuing delay due to the link bandwidth. These assumptions are easily justified for tests involving ping or other low-bandwidth probing.

From the figure we have that the pth packet is generated at time

$$t_p = pT_p \tag{3}$$

and the sth transmission slot is at time

$$t_s = sT_s + \Phi_s. \tag{4}$$

At time t_p, when the packet is generated, the next available timeslot, when the packet is transmitted, is

$$s = \left\lceil \frac{pT_p - \Phi_s}{T_s} \right\rceil. \tag{5}$$

Subtracting (4) from (3) and combining with (5) gives the packet delay:

$$t_d[p] = t_s - t_p = \left(\left\lceil \frac{pT_p}{T_s} - \frac{\Phi_s}{T_s} \right\rceil + \frac{\Phi_s}{T_s} \right) T_s - pT_p. \tag{6}$$

We next define the ratio between T_s and T_p as

$$\frac{T_s}{T_p} = \frac{n}{k}, \tag{7}$$

where n and k are coprime natural numbers. We also define the normalized slot phase as

$$\varphi = \frac{\Phi_s}{T_s}, \tag{8}$$

where φ represents the fraction of the slot period by which the first slot is shifted relative to the time when the first packet is generated. We can now rewrite (6) as

$$t_d[p] = \left(\left\lceil p\frac{k}{n} - \varphi \right\rceil + \varphi \right) \frac{n}{k} T_p - pT_p, \tag{9}$$

which can be easily rewritten as (1).

Two special cases are of interest. For $n = 1$ and $k \in N$, the slot rate is an integer multiple of (or equal to) the packet-generation rate and the phase difference between the packet-generation and link-slot grids is constant and equal to Φ_s. Because $0 \le \varphi \le 1$ and kp is an integer, we have

$$\left\lceil kp - \varphi \right\rceil = kp \tag{10}$$

and (1) reduces to Φ_s. In other words, the delay is constant, but ambiguous; for each experiment, we observe a different delay for the duration of the experiment that can be anywhere between zero and T_s. This is essentially the result shown in Fig. 4.

For $k = 1$ and $n > 1$, multiple consecutive packets are mapped to the same transmission slot. Per our assumption, there is enough bandwidth for all packets to be transmitted in that slot; however, the packets generated earlier for a given slot will incur higher delay than the packets generated later for that same slot. The slot s in which a given packet p is transmitted is given by

$$s = \left\lceil \frac{p}{n} - \varphi \right\rceil, \tag{11}$$

and hence for a given slot s, all packets that satisfy

$$(s - 1 + \varphi)n < p \le (s + \varphi)n \tag{12}$$

are transmitted in s. Substituting (8), (11), and $k = 1$ into (1) gives the following value for packet delay,

$$t_d[p] = (sn - p)T_p + \Phi_s, \tag{13}$$

where s is given by (11). The graph of this function is a sawtooth with amplitude $nT_p = T_s$ and bias (i.e., y offset) Φ_s. Similarly to the previous case, the bias will remain constant throughout the test, but when the test is repeated, the bias will change.

The last and the general case happens when $k \ne n$ and $k, n > 1$. In this case the packet-generation grid slides in time relative to the link-slot grid. The general expression for packet delay (1) can produce complex sawtooth patterns depending on the specific values of n and k.

An important difference between the sawtooth patterns produced in this case compared to the previous cases is that in this case the bias is zero. Because the two grids slide in time, eventually a packet will be generated immediately before the transmission slot, incurring zero delay.

In the measurement of Fig. 3 $k = 399$, $n = 400$, and the bias shown in the figure does not come from the slotted link. This bias is inherent to the transmission path and consists of the sum of all propagation delays (fibers, air, and hardware and software paths).

To model the delay through the software stack — that is, the time it takes to propagate the packet from the application socket to the UE modem — we rewrite (5) as

$$s = \left\lceil \frac{pT_\mathrm{p} + \delta T_\mathrm{s} - \Phi_\mathrm{s}}{T_\mathrm{s}} \right\rceil, \tag{14}$$

where δ is a dimensionless random variable expressed as a fraction of T_s. Plugging (14) back into (4) and reworking the derivations yields (2). Note that we do not add δ-term to (3), because the ping utility calculates the measured delay as the difference between the packet arrival time and the time when the packet was placed on the socket. In other words, the application is not aware of the delay through the software stack and does not account for it.

REFERENCES

[1] 2013. Timing and Synchronization for LTE-TDD and LTE-Advanced Mobile Networks. Symmetricom White Paper. (2013).

[2] 2016. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification (3GPP TS 36.321 version 13.0.0 Release 13). ETSI TS 136 321 v13.0.0.0. (Feb 2016).

[3] 2016. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures (3GPP TS 36.213 version 13.0.0 Release 13). ETSI TS 136 213 v13.0.0.0. (May 2016).

[4] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar. 2014. GENI: A Federated Testbed for Innovative Network Experiments. *Comput. Netw.* 61 (Mar 2014), 5–23.

[5] C.-Y. Chang, K. Alexandris, N. Nikaein, K. Katsalis, and T. Spyropoulos. 2016. MEC Architectural Implications for LTE/LTE-A Networks. In *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture (MobiArch '16)*. ACM, 13–18.

[6] J. Fabini and M. Abmayer. 2013. Delay Measurement Methodology Revisited: Time-Slotted Randomness Cancellation. *IEEE Trans. Instrum. Meas.* 62, 10 (Oct. 2013), 2839–2848.

[7] J. Fabini and T. Zseby. 2016. The Right Time: Reducing Effective End-to-End Delay in Time-Slotted Packet-Switched Networks. *IEEE/ACM Trans. Netw.* 24, 4 (Aug. 2016), 2251–2264.

[8] A. Gosain, M. Berman, C. Li, Y. Wang, and H. Jin. 2016. Enabling Campus Edge Computing using GENI Racks and Mobile Resources. In *Proc. IEEE/ACM Symposium on Edge Computing*. ACM, 41–50.

[9] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet. 2017. Low latency MEC framework for SDN-based LTE/LTE-A networks. In *Proc. IEEE Int. Conf. Communications (ICC)*.

[10] International Telecommunication Union (ITU), Telecommunication Standardization Sector 2013. *Timing and Synchronization Aspects in Packet Networks*. International Telecommunication Union (ITU), Telecommunication Standardization Sector. ITU-T Recommendation G.8261.

[11] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. 2015. Edge-centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev.* 45, 5 (Oct. 2015), 37–42.

[12] Y. Mao, J. Zhang, and K Letaief. 2016. Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. *IEEE J. Sel. Areas Commun.* 34, 12 (Dec 2016), 3590–3605.

[13] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, and J. V. Merwe. 2014. Towards Understanding TCP Performance on LTE/EPC Mobile Networks. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, &#38; Challenges*. ACM, 41–46.

[14] OAI 2017. Open Air Interface — 5G software alliance for democratising wireless innovation. Online. (2017). http://www.openairinterface.org/.

[15] Patel, et *al.* 2014. Mobile-Edge Computing – Introductory Technical White Paper. European Telecommunications Standards Institute (ETSI). (Sep 2014).

[16] M. Satyanarayanan. 2014. A Brief History of Cloud Offload. *ACM/SIGMOBILE GetMobile Magazine* 18, 4 (Apr 2014), 19–23.

[17] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4 (Oct-Dec 2009), 2–11.

[18] Time Machines by CSS 2016. *TM2000A PTP and NTP Time Server GPS Time Sourced – Installation and Operation Manual* (rev. h 11-8-2016 ed.). Time Machines by CSS. http://www.css-timemachines.com/wp-content/uploads/TM2000AManual.pdf.

[19] Y. Yu. 2017. Mobile Edge Computing Towards 5G: Vision, Recent Progress, and Open Challenges. *China Communications* 13, Supp. 2 (Jan 2017), 89–99.