P4EC: Enabling Terabit Edge Computing in Enterprise 4G LTE

Max Hollingsworth Jinsung Lee Zhang Liu Jihoon Lee Sangtae Ha
Dirk Grunwald
Department of Computer Science, University of Colorado Boulder

Abstract

Traditional LTE networks route Internet traffic through a packet gateway. Enterprise LTE networks with a cloud-based core use a similarly faraway gateway. To provide low-latency services, such as accessing nearby mobile devices, fog services, or localized information, a "local-exit" to the Internet is needed to avoid traveling through the LTE core. To create a local-exit, we build P4EC, a terabit capable mobile edge cloud using a programmable switch to distinguish and reroute traffic. P4EC is placed physically near the cellular deployment and reroutes specifically identified traffic to and from the mobile device. The P4EC implements packet redirection using the P4 programmable switching hardware that supports terabit throughput in inexpensive equipment. P4EC operates without any modification to the LTE core. This work describes a working proof-of-concept operating in an actual LTE network.

1 Introduction

LTE networks have three major components: mobile devices (UEs), the radio access network composed of base stations (eNBs) and the core network (EPC). The EPC serves a variety of functions including subscriber authorization, managing UE movement, and provides a packet gateway (PGW) to the Internet. A large mobile network operator (MNO), such as AT&T, typically uses one EPC to serve a very large physical region, sometimes many states. All cellular traffic in that region passes through the PGW on its way to the Internet.

Currently, there are only a handful of MNOs operating in the United States. The number of MNOs is likely to grow rapidly due to recent changes in spectrum policy. In 2012, the FCC allocated 150 MHz of spectrum in the 3.5 GHz band for use by private LTE operators using the Citizens Broadband Radio Service (CBRS) [12]. In September of 2019, the FCC approved the first commercial deployments of the CBRS access system [13]. When coupled with License Assisted Access (LAA) operation in 5 GHz band, there will likely be many small-scale network operators. CBRS and LAA are particularly useful for enterprise, campus, and municipality scale LTE deployments.

With this increase in the number of small-scale LTE operators comes an opportunity for cloud providers, like Amazon, to offer a cloud-EPC [8]. A cloud-EPC model would mean purchasing physical eNBs (similar in size to a WiFi router) and deploying them over a property (e.g., a warehouse or campus) with an Internet backhaul to connect to a cloud-EPC service. The cloud-EPC simplifies the deployment for the enterprise CBRS LTE operator. But, it also binds them to the latency created by having a PGW in a distant cloud datacenter.

There are many latency sensitive applications that rely on LTE (e.g., augmented reality, wearable cognitive assistants, and IoT sensors). Latency is most crucial in public safety applications. Command and control applications used by fire departments are a good example of a localized, time-sensitive service that cannot rely on a distant cloud datacenter. These applications provide firefighters with critical information such as orders, location, and movement. Our results show large delay reductions ($\geq$100ms) in video streaming applications similar to those deployed by public safety. CBRS operators would be challenged to provide public safety members with low-latency services, such as these, simply because of the distance of the cloud-EPC.

Our solution to the latency issues caused by the cloud-EPC is the P4EC. When deployed, the P4EC acts as a middle-box, monitoring all traffic between the eNB and EPC. P4EC, shown in Figure 1, is a programmable P4 switch using the Barefoot Tofino chipset with multiple 100 GbE ports. The Tofino switch provides line-rate packet processing with an aggregate 2.0 Tbps throughput.

At this throughput the P4EC switch can handle the combined traffic from over 7,300 eNBs[1] [5]. Each eNB can serve hundreds of active UEs. The switch costs slightly more than $9,000 making the price per active UE roughly a penny.

Using custom P4 programs, traffic deemed latency sensitive is redirected to the "local-exit". The P4EC serves two roles

[1] An eNB is assumed to support a single-sector, 20 MHz FDD, 2x2 MIMO, 256-QAM downlink, and 64-QAM uplink.

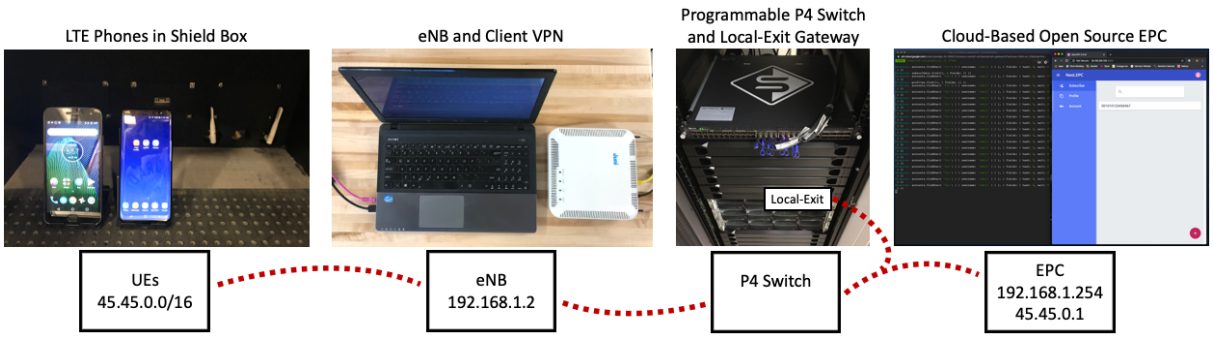

Figure 1: Our end-to-end LTE testbed consists of UEs, an eNB, a P4 programmable switch (P4EC), and a cloud-EPC. The switch reroutes traffic meant for low-latency services out the local-exit.

in an LTE deployment; first, it provides a quick access to fog-computing resources physically near the eNB, thus offering faster round-trip communication than the cloud. Second, the P4EC acts as a "bump-in-the-wire" for non-fog-bound traffic and control information passing between the eNB and the EPC. As a result, the P4EC can function without any modification to the EPC. Thus P4EC is an easily added, inexpensive system ready for large-scale LTE deployments.

2 Related Work

Providing edge computing in mobile networks (Mobile Edge Computing, or MEC) has been a popular topic in recent years. The following explains research and standardization efforts for enabling MEC in 4G/5G networks.

Architecture proposals: Several works have studied the architectural design for MEC deployments over cellular networks. Chang et al. [10] proposed a standard-compliant modular MEC architecture and described the functional mapping to LTE systems. Huang et al. [16] presented an OpenFlow-based MEC framework that can provide the required data-plane flexibility and programmability. Huang et al. [17] presented a prototype implementation of a MEC platform by developing an application-aware traffic redirection mechanism at the edge network. However, all these solutions require modifications on the eNB and the existing core network, so large-scale deployment can be challenging.

A middlebox approach [20] simplifies deployment of a MEC platform in 4G LTE networks. Their work is standard-compliant and transparent to existing network components for deployment in the field. We recognized the value of the proposed transparent middlebox but have identified a few limitations: a) packet-filtering and forwarding, performed on a PC with Python and iptables, is not a commercial solution; b) there is no mechanism to distinguish *privileged UEs* that should have access to edge services; c) access is provided only to an edge machine; and, d) redirection will not work with DNS over HTTPS, which is being rapidly deployed. To address these limitations, this work a) implements a P4 switch solution capable of handling terabits of traffic at line-rate; b) identifies the UE through source IP address; c) provides a local-exit; and d) performs transport-layer redirection. The local-exit means that the UE can access fog services, nearby UEs on different providers, and the Internet. Because our solution is programmable, it can be extended to include different access control policies, encapsulation methods, and transport options.

MEC in 4G LTE: Traffic offloading in 4G LTE has been used for reducing backhaul traffic [1, 2]. LIPA (Local IP Access) considers a limited network architecture where a small cell eNB has a local gateway functionality and enables the UE to access other IP capable devices in the same IP network without traversing the core network. This feature only supports local connection between devices in physical proximity. SIPTO (Selective IP Traffic Offload) assumes a similar network architecture and enables the UE connected via the eNB to access a defined IP network (e.g., Internet) directly without traversing the PGW. SIPTO is a network-driven solution and cannot be controlled by the UE. More importantly, these proposals have not been specified in the 3GPP standards and also not deployed in the field.

5G standard efforts: Apart from 4G EPC architecture, 5G network standards have included several features to facilitate MEC deployment for operators and 3rd party services [3, 4]. Specifically, 5G core allows distributed deployment of multiple gateways, so that it can select a proper gateway close to the UE's point of attachment, based on which various advanced features for edge computing can be realized. For example, 5G core can accommodate the concept of a local area data network that provides an isolated data service to UEs within a certain geographic area [19]. In addition, there is an ongoing effort to integrate such MEC standards within the 5G network for efficient management and orchestration of edge computing applications [11].

3 System

There are a number of design decisions needed to create the P4EC. Here, we describe two of the more significant decisions: content redirection and transparency to the EPC. As a

note, other design decisions, such as how to identify UEs and routing rules, are described in the implementation section.

Content Redirection: The first design decision is how to identify content (e.g., a TCP session, or real time streaming) to reroute *via* the local exit. This practice, called *content redirection*, is commonly used by CDNs and comes in three general techniques: DNS redirection, transport-layer redirection, and application-layer redirection [7]. Based on the following analysis of these three techniques, we implemented transport-layer redirection in P4EC.

DNS redirection, as used by CDNs, employs a name server to respond to the client with an A (address) record of a surrogate server rather than the content originator. DNS redirection has been proposed for use in other LTE MEC designs, and if deployed in the P4EC would intercept all DNS queries and respond only to domains that it recognizes and forward the rest. Using DNS redirection can ensure an entire session is rerouted and that the content goes to the appropriate fog server, given network load, content availability, and proximity. The drawbacks include: a) DNS only allows for resolution at the domain level, where a redirect at the per object level might be desired, and b) secured DNS techniques like DNS-over-HTTPS and DNSSEC make DNS redirection impossible in middleboxes.

Transport-layer redirection deploys an in-path element to examine a packet's IP addresses, ports, and protocol to decide whether to reroute. When a packet is flagged, it is rerouted to a virtual surrogate that acts as the end point, this is labeled as NAT in Figure 2. The virtual surrogate, who has knowledge of the edge network's topology and load, forwards to another surrogate or group of surrogates to handle the actual service. The benefits of transport-layer redirection are that the P4EC does not need to know about the edge network's status, i.e., surrogate load, availability, proximity, and bandwidth. The P4EC is only aware of the virtual surrogate, and the edge service provider that exists past the NAT has control of load balancing across the edge. NAT services can either be implemented in a separate system or implemented at terabit speeds using P4; our prototype uses a separate NAT middlebox.

There are multiple methods to implement application-level redirection which uses an in-path element that parses application-layer headers for fields such as URL, cookies, language, and user-agent in order to select an appropriate surrogate. This provides the benefit of fine-grained redirection control of an individual object. Unfortunately, without application level coordination, TLS encryption obscures most of the application-layer information of interest.

For P4EC, we chose transport-layer redirection because an edge service provider can define the exact behavior based on their knowledge of surrogate load, surrogate availability, proximity, bandwidth, and content availability. This changes the P4EC's primary role to classifying traffic and incorporating edge information to effect load balancing. We see this simplification of the P4EC's role as an advantage.

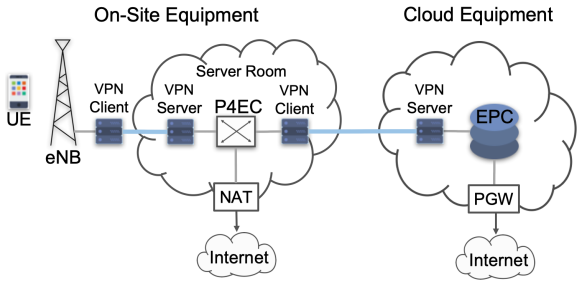

Figure 2: A network diagram of our LTE entities: UE, eNB, P4EC, and cloud-EPC. The on-site equipment is located within our research center. And the EPC is hosted in the cloud in the nearest Google data center.

Transparency: The P4EC can either operate with or without EPC coordination. There are three different levels of transparency we considered. The first, non-transparent method, involves modifying the EPC to notify the P4EC when eligible UEs attach to the network. This method simplifies the P4EC's responsibilities but could only be deployed with modified-EPCs. The second method is to monitor the control messages (S1AP) that pass between the eNB and EPC. When the P4EC sees a UE attach message it records the UE's identifier, MME-UE-S1AP-ID, and will serve that UE until the detach message is seen. The attach request message contains the UE's unique identifier called IMSI (International Mobile Subscriber Identity). The network operator must specify the IMSIs of the UE's permitted to use local exits. This method is transparent and involves no modification of the EPC, but will not work with temporary identifiers called Temporary Mobile Subscriber Identity (TMSI). The third method involves no modification to the EPC and no operator interaction with the P4EC after installation. This method relies on the UE to self-identify to the P4EC with out-of-band signals (authorization packets). These packets would contain credentials that notify the P4EC that the UE is an approved user. The drawback to this method is that each application that wants a local-exit access must self-identify to the P4EC. We believe this will be our ultimate solution though we currently have the second method implemented.

4 Implementation

Testbed setup: The UE is a Galaxy S8. The eNB, a Juni LTE Small Cell [18], is physically located in our lab. When powered on, it establishes a connection with the EPC; we use the open-source NextEPC [22] that we deploy on Google Cloud Platform (GCP) [14]. The P4EC runs on a Stordis BF2556x-1T [25] located in a server rack in the same facilities as our lab. Figure 2 shows the network diagram of the four LTE entities: UE, eNB, P4EC, and EPC. The eNBs and the EPC typically exist on a private IP network. To create a private network between the eNB and cloud-EPC we use OpenVPN [23]; al-

though IPSEC can be implemented in P4, we have not taken that step.

Cloud-EPC setup: There are two concerns when setting up a cloud-EPC. The first is security and the second is successfully routing packets from the eNB to the EPC and back. Many installations use a VPN or IPSEC for security and the GPRS Tunnel Protocol (GTP) is used to tunnel a UE's data between the eNB and EPC, as specified in [6]. Assuming the EPC has a public IP address and the eNB is on a private network behind a NAT, outgoing UE traffic will reach the EPC but the return traffic cannot reach the UE because of the private network and tunnel. We use a VPN to give the EPC access to the eNB's private network.

The cloud VPN server and EPC are implemented using an Ubuntu 16.04 server with four virtual CPUs and 16 GB of memory. This is sufficient to handle the traffic generated by the UEs in our latency tests. The cloud-EPC can easily scale by deploying larger VM instances.

P4 switch: The P4EC is a P4 programmable switch that performs routing to and from the local-exit. P4 [9] is a high level programming language for packet processing, which allows protocol independent, deep programmability for the data plane using Match-Action tables. The P4 Tofino packet processing pipeline contains the following components: Parsers, Deparsers, Ingress processing, Packet Replication Engine (PRE) and Egress processing. Match-Action tables are located inside Ingress/Egress Processing. PRE is located between Ingress/Egress processing and is used for advanced functions such as Packet Mirroring, Multi-Cast, etc. Parsers and Deparsers are used to parse and reconstruct packet headers before and after Ingress/Egress processing. Once the data plane logic is described using P4 language, a special compiler is then used to compile P4 into Tofino Native Architecture binary that can be run on our switch.

The P4EC monitors traffic between the eNB and the EPC. This includes all of the control signalling (S1AP) and the data (GTP) traveling to and from the UEs. By monitoring the S1AP packets for Attach-Request and Detach messages the P4EC maintains a table of the active UEs. Actions for specific packets (e.g., a flow from a UE) are specified in *tables* and perform matching at wire speed. A controller processor on the P4 switch populates the re-routing tables using either an out-of-band API or by reacting to authorization packets from the UE. Unmatched packets are routed to the EPC.

In order to reroute packets to an edge server, the P4EC matches on the UE's IP and the destination IP address. Then the decapsulation stage removes the outer IPv4, outer UDP, and GTP headers. The P4EC modifies the out-going MAC and optionally the source address if performing NAT.

For a return packet from the edge to the UE, the P4EC must inject the packet back into a GTP tunnel. First, a match occurs on the destination IP, which should be the UE's IP. Then the packet is encapsulated with appropriate IPv4, UDP, and GTP headers and is sent to the eNB. These headers mimic the packets coming from the EPC, and the GTP tunnel identifiers are tracked by monitoring S1AP packets.

5 Evaluation

To inspect the latency of the separate stages of the LTE network we use the ping tool. As shown in Figure 3, we perform separate latency tests from the UE to the two Internet gateways at its disposal: PGW and NAT. The test on the left shows the UE to cloud-PGW round-trip time (RTT), which is separated into time spent in the radio access network (RAN) and passing between the eNB and PGW (network core). The test on the right shows the UE to NAT RTT, which is also separated into time spent in the RAN and network core.

The measurements were taken with 300 pings to the PGW and NAT each. We use the same RAN in both tests. We find that the RAN introduces much more latency variance than the network core even with single packet pings. Waiting time may vary depending on when a packet becomes ready, which has to be scheduled in the next uplink transmission period [24]. Note that our latency measurements on the RAN side are consistent with recent 4G/5G measurements [21].

The local-exit's success in decreasing overall latency is shown by the "Network Core" columns. The "Network Core" columns are highlighted in Figure 4. As the PGW is physically located at a distance (about 550 miles away in this case), it is expected that there is some latency in RTT. For this test we chose the GCP region nearest to our facilities. The added RTT is about 13ms. Traffic that passes through the PGW does not only suffer from added latency, but is also exposed to increased variance of the public Internet (indicated by the heavy tail in the "eNB - PGW" column).

From Figure 5 we show the advantage that the local-exit has over the nearest datacenter and other datacenters increasingly far away. This highlights that the local-exit's efficacy can be amplified in situations when the cloud-EPC becomes farther from the eNB. The local-exit offers flexibility to the network operator who may decide to deploy the cloud-EPC in a sub-optimal location for cost, convenience, etc.

5.1 Application Performance

Web browsing: We use web browsing as a simple but real-world application to test the performance of the P4EC. On the UE we run curl to repeatedly measure the total time to download a webpage. The download is 100KBs in size and the UE and server exchange about 150 TCP packets. The webpage is hosted geographically near our facilities and was specifically chosen to show off the advantage of having a local-exit near the eNB. It is important to note that P4EC allows us to enable the local-exit for a specific UE and a specific set of destination addresses – we match on the 5-tuple (protocol, src address and port, dst address and port) and other UEs using the same network are not redirected.

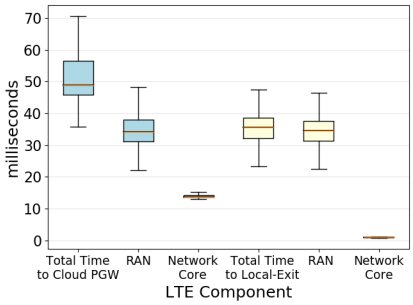

Figure 3: Ping RTT from UE to the cloud-PGW vs. from UE to the local-exit. Also, separated into time within RAN and core.

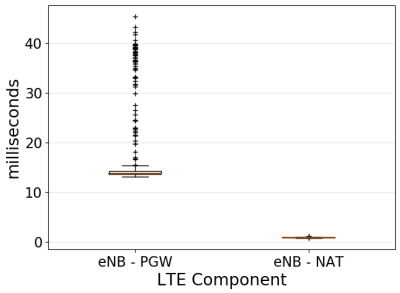

Figure 4: Ping RTT from the eNB to the cloud-PGW vs. from the eNB to the local-exit.

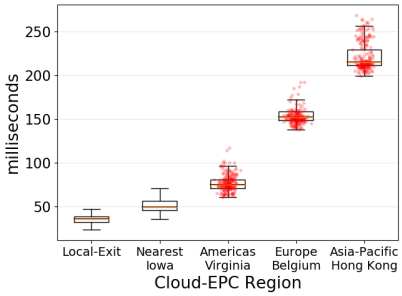

Figure 5: Ping RTT from the UE to the local-exit vs. from the UE to the PGW of increasingly distant datacenters.

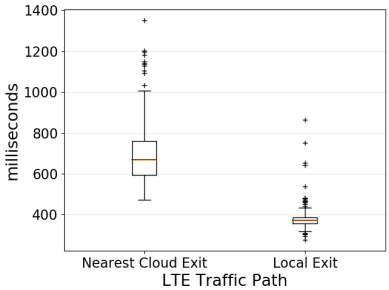

Figure 6: Webpage download time using local-exit vs. the nearest cloud exit.

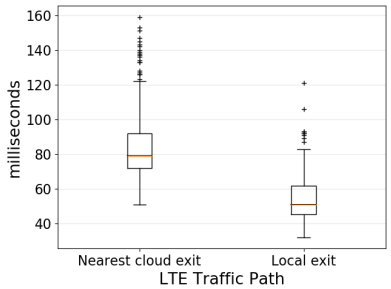

Figure 7: WebRTC RTT for 4 minute streaming video call.

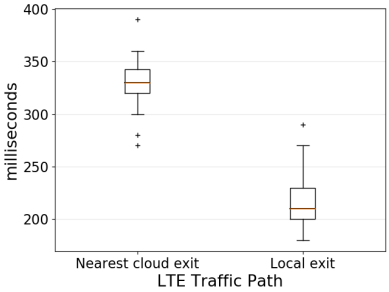

Figure 8: WebRTC streaming delay for 4 minute streaming video call.

Figure 6 shows that the median latency fell nearly 300ms and the variance greatly narrows. By using the local-exit, we avoid the 13ms round-trip to the PGW and back, twice. But 26ms falls short of explaining the 300ms of savings. The cause is likely the compounding effects of buffering at routers and other middleboxes in the Internet, with the impact of waiting for TCP ACKs to travel and the extra 26ms before continued transmission. The latency will be further reduced when the P4EC performs network translation.

Live video streaming: We use the WebRTC [15] application to evaluate live video streaming. The video calls are established between a Galaxy S8 connected to our LTE testbed and a high-end laptop (running Ubuntu 18.04) connected to the Internet via campus Ethernet.

WebRTC is configured with the resolution of 1280x720 (HD) at 30fps and the max video encoding rate of 2.5 Mbps. For the evaluation, we measure RTT at the transport layer and end-to-end streaming delay, defined as the time lag from when a video frame is generated on the sender until it is displayed on the receiver's screen. Thus, it consists of video capturing, encoding, transmission, decoding, and rendering delays. We follow the measurement methodology of a previous measurement study [26].

Figure 7 shows the transport layer RTT for WebRTC. We compare the video call through the cloud-EPC to the video call through the local-exit. The local-exit call has a median RTT 27ms faster than the cloud-EPC. This is explained by the 26ms reduction in travel by avoiding the cloud-EPC. The end-to-end streaming delay, shown in Figure 8, also compares the local-exit to the cloud-EPC. The local-exit causes a large reduction in delay, more than 100ms. WebRTC video streaming is a good analog for applications that benefit from a local-exit. Already, there are scenarios when low-latency streaming between two physically near parties is essential such as public-safety members responding to an emergency.

6 Conclusion and Future Work

In this paper we presented P4EC, a mechanism for terabit capable mobile edge cloud in enterprize-scale LTE networks. We use a P4 programmable switch to distinguish and reroute traffic. The Tofino platform is malleable enough to cover many applications and the hardware is powerful enough for commercial use.

We measure P4EC against web browsing and live video calling. In both cases we saw significant decreases in latency. Our proof-of-concept implementation showed that the P4 switch is a viable, inexpensive candidate for P4EC, and that the P4EC can function without any modification to the network core.

As our future work, we would like to answer the following questions: (i) How can we maintain edge services as a UE moves from one eNB to another? (ii) How do we identify the UE as a subscriber (when TMSI changes regularly)? (iii) Can the UE self-identify as P4EC-eligible? (iv) Is P4EC interoperable with other open-source EPCs?

7 Discussion Topics

There are two sets of problems in this paper. One set of problems is deciding what LTE model makes sense for CBRS deployments (i.e., a cloud-EPC vs. local-EPC vs. a hybrid solution). The other set of problems is given a specific CBRS model, how do we deliver high-performance edge computing to those deployments.

We are interested in the audience's feedback on what CBRS LTE (and 5G) deployments will look like. We have assumed CBRS operator will subscribe to a cloud-EPC service and deploy eNBs (and SIM cards) around their facilities. But other ideas include a cloud-EPC with the MME on-site to manage the eNBs and the movement of UEs amongst them.

We would also like to address the implementation specific problems of our system, of which there are a few: is it realistic to avoid EPC modification to have a working MEC switch? How does one passively identify a UE when their TMSI changes regularly? How do you deter users from abusing the local-exit system? How do we maintain edge services while the user is moving from one eNB to another? What are some specific latency requirements of various edge-dependent applications?

References

[1] 3GPP. Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO) (Release 10), 2011. http://www.3gpp.org/dynareport/23829.htm.

[2] 3GPP. Local IP access (LIPA) mobility and Selected IP Traffic Offload (SIPTO) at the local network (Release 12), 2013. http://www.3gpp.org/dynareport/23859.htm.

[3] 3GPP. 5G; Procedures for the 5G System (5GS) (3GPP TS 23.502 version 15.8.0 Release 15), 2020. http://www.3gpp.org/dynareport/23502.htm.

[4] 3GPP. 5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 15.8.0 Release 15), 2020. http://www.3gpp.org/dynareport/23501.htm.

[5] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (3GPP TS 36.213 version 16.1.0 Release 16), 2020. http://www.3gpp.org/dynareport/36213.htm.

[6] 3GPP. LTE; General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access (3GPP TS 23.401 version 15.10.0 Release 15), 2020. http://www.3gpp.org/dynareport/23401.htm.

[7] R. Nair O. Spatscheck A. Barbir, B. Cain. Known content network (cn) request-routing mechanisms. RFC 3568, 7 2003.

[8] Monica Alleven. Federated launches new CBRS services with AWS, Microsoft Azure, Feb 2020. http://www.fiercewireless.com/wireless/federated-launches-new-cbrs-services-aws-microsoft-azure.

[9] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

[10] Chia-Yu Chang, Konstantinos Alexandris, Navid Nikaein, Kostas Katsalis, and Thrasyvoulos Spyropoulos. MEC Architectural Implications for LTE/LTE-A Networks. In *Proceedings of ACM MobiArch*, 2016.

[11] Sami Kekki et al. MEC in 5G networks. White Paper 28, ETSI, June 2018. https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf.

[12] FCC. Amendment of the Commission's Rules with Regard to Commercial Operations in the 3550-3650 MHz Band, 2012. http://transition.fcc.gov/Daily_Releases/Daily_Business/2012/db1212/FCC-12-148A1.pdf.

[13] FCC. WTB and OET Approve Initial Commercial Deployments in 3.5 GHz Band, 2019. https://www.fcc.gov/document/wtb-oet-approve-initial-commercial-deployments-35-ghz-band.

[14] Google. Cloud Computing Services. https://cloud.google.com, 2020.

[15] Google. WebRTC. https://webrtc.org, 2020.

[16] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet. Low latency MEC framework for SDN-based LTE/LTE-A networks. In *Proceedings of IEEE ICC*, 2017.

[17] S. Huang, Y. Luo, B. Chen, Y. Chung, and J. Chou. Application-Aware Traffic Redirection: A Mobile Edge Computing Implementation Toward Future 5G Networks. In *Proceedings of IEEE International Symposium on Cloud and Service Computing (SC2)*, 2017.

[18] Juni. Enterprise Small Cell JL620. http://vidatele.com/smallcellsolutions.html, 2017.

[19] J. Lee, S. Moon, B. Bae, and J. Lee. Local Area Data Network for 5G System Architecture. In *Proceedings of IEEE 5G World Forum (5GWF)*, 2018.

[20] Chi-Yu Li, Hsueh-Yang Liu, Po-Hao Huang, Hsu-Tung Chien, Guan-Hua Tu, Pei-Yuan Hong, and Ying-Dar Lin. Mobile edge computing platform deployment in 4g LTE networks: A middlebox approach. In *Proceedings of USENIX HotEdge 18*, 2018.

[21] Arvind Narayanan, Jason Carpenter, Eman Ramadan, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. A First Measurement Study of Commercial mmWave 5G Performance on Smartphones, 2019.

[22] NextEPC Inc. Open source implementation of LTE EPC. https://nextepc.org, 2019.

[23] OpenVPN. VPN Software Solutions and Services. https://openvpn.net, 2020.

[24] Shinik Park, Jinsung Lee, Junseon Kim, Jihoon Lee, Sangtae Ha, and Kyunghan Lee. ExLL: An Extremely Low-latency Congestion Control for Mobile Cellular Networks. In *Proceedings of ACM CoNEXT*, 2018.

[25] Stordis. STORDIS BF2556X-1T. https://www.stordis.com/products/stordis-bf2556x-1t/, 2019.

[26] C. Yu, Y. Xu, B. Liu, and Y. Liu. "Can you SEE me now?" A measurement study of mobile video calls. In *Proceedings of IEEE Infocom*, 2014.