Mobile Edge Computing Platform Deployment in 4G LTE Networks: A Middlebox Approach

Chi-Yu Li[1], Hsueh-Yang Liu[1], Po-Hao Huang[1], Hsu-Tung Chien[1],
Guan-Hua Tu[2], Pei-Yuan Hong[1], and Ying-Dar Lin[1]

[1]Department of Computer Science, National Chiao Tung University
[2]Department of Computer Science and Engineering, Michigan State University

Abstract

Low-latency demands for cellular networks have attracted much attention. Mobile edge computing (MEC), which deploys a cloud computing platform at the edge closer to mobile users, has been introduced as an enabler of low-latency performance in 4G and 5G networks. In this paper, we propose an MEC platform deployment solution in 4G LTE networks using a middlebox approach. It is standard-compliant and transparent to existing cellular network components, so they need not be modified. The MEC middlebox sits on the S1 interface, which connects an LTE base station to its core network, and does traffic filtering, manipulation and forwarding. It enables the MEC service for mobile users by hosting application servers. Such middlebox approach can save deployment cost and be easy to install. It is different from other studies that require modifications on base stations or/and core networks. We have confirmed its viability through a prototype based on the OpenAirInterface cellular platform.

1 Introduction

Mobile edge computing (MEC) is one key technology to achieve low-latency performance in cellular networks. It has been determined as a key feature in future 5G networks by both ETSI [11] and 3GPP [5] standardization organizations[1]. It seeks to provide a cloud computing platform at the network edge to be closer to mobile users than conventional cloud systems. It can reduce service latency with two major merits. First, short end-to-end distance leads to small propagation delay and avoids bandwidth bottleneck or congestion on the Internet. Second, offloading services from the cloud, which may have congestion, to the edge reduces computation delay. Due to emerging low-latency demands, several MEC deployment solutions are being in development [13].

[1]ETSI has dropped the 'Mobile' out of MEC and renamed it as Multi-access Edge Computing (MEC) since 2016 [18].

In this paper, we seek to design an MEC platform that can be easily deployed in 4G LTE networks, as well as may be used as a reference design for future 5G networks. Though ETSI has proposed several MEC deployment options [13] and introduced its reference architecture [12], it mainly poses requirements and possible issues, but does not have concrete designs or implementations. In addition, several proposed solutions [10, 14, 15] are not compliant to the 3GPP standards because of their modification requirements on the 4G base stations or/and core networks. They may cause large deployment costs and impede deployment incentives.

We adopt a middlebox approach to develop the MEC platform. Installing it on an LTE network only needs to connect the LTE base station (eNB) and the serving gateway (S-GW) to it with network cables, as well as then configure it. No modifications are required on the connected eNB and S-GW. It works for any standard-compliant network infrastructure without any upgrade of network components. Both cellular network carriers and vendors can benefit from this approach. For the carrier, it is a flexible, low-cost solution. Its MEC deployment needs only an MEC middlebox, and is not restricted to its existing infrastructure vendors, which may have poor or expensive support of the MEC. For the vendor, it can be a standalone solution which can be offered to new customers, which are not using the vendor's infrastructure.

As a network middlebox, the MEC platform performs traffic filtering, manipulation and forwarding over an S1 interface, which connects the eNB and the S-GW. The major challenge is how to deal with GPRS tunneling protocol (GTP) tunnels, which carry data traffic along the S1, and do traffic redirection to enable MEC applications. To this end, we design the MEC middlebox based on four major ideas: address resolution protocol (ARP) proxy, GTP repackaging, traffic redirection via DNS, and stateful tracking of GTP tunnels. We further prototype it with the OpenAirInterface (OAI) cellular platform [1], and examine the performance of web and video stream-

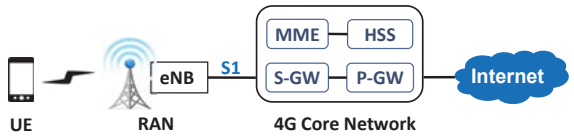


Figure 1: 4G LTE network architecture.

ing services. Compared with the conventional servers on the Internet, our MEC platform can shorten the web service's median latency and the video service's 95th jitter with up to 69.86% and 67.85%, respectively.

2 Background and Related Work

4G LTE Network. The 4G LTE network architecture as shown in Figure 1 contains three main components: core network, radio access network (RAN) and user equipment (UE). The UE accesses the Internet through the other two components. There are control and user planes spanning them. The former takes care of mobility, security and resource reservation functions. The latter routes traffic between the UE and the Internet. The MEC is a network concept that enables computing capabilities and service supply at the edge next to the RAN.

In the core network, the control plane includes two main entities, mobility management entity (MME) and home subscriber server (HSS). The MME administrates mobility management, security services and resource reservation, whereas the HSS maintains user subscription and security contexts. The user plane spans serving gateway (S-GW) and packet data network gateway (P-GW), so data traffic between the UE and Internet hosts traverses them. The RAN consists of LTE base stations, which are called evolved Node B (eNB).

There exists an S1 interface [6] between the eNB and the core network, and it consists of two parts: S1-MME [8] and S1-U [7] interfaces. The S1-MME interface is responsible for the control plane to deliver signaling messages between the eNB and the MME. The S1-U interface delivers data traffic between the eNB and the S-GW. The transport relies on GPRS tunneling protocol (GTP) [3], where one GTP tunnel is built for each UE's Internet traffic. The UE's IP packets are thus carried by GTP tunnels on the S1-U interface. Note that the GTP tunnels are built on top of UDP/IP transport, which is used for the internal communication between LTE network elements (e.g., eNB, MME, S-GW, P-GW, etc.), but not the one between the UE and Internet hosts.

Related Work. They have been several surveys [9, 19, 16, 17] that focus on the MEC from various perspectives. Ahmed *et al.* [9] present some promising MEC application scenarios, as well as discuss their key attributes and research challenges. Taleb *et al.* [19] study enabling technologies in the MEC, which include network virtualization, slicing and orchestration, as well as software defined networking (SDN). Mach *et al.* [16] focus on computation offloading problems by considering offloading decision, computing resource allocation, and mobility management. Mao *et al.* [17] do a comprehensive literature review on joint radio and computation resource allocation for the MEC. However, neither of them examine the MEC deployment issue in the existing 4G or future 5G networks.

There have been several research studies [10, 14, 15] proposing solutions for the MEC deployment in LTE networks. The first two studies [10, 14] address how the MEC cooperates with LTE network elements (e.g., eNB, P-GW, MME, etc.), and interconnects them using new defined interfaces or an SDN architecture. The last one [15] deploys the MEC inside the eNB by redirecting the UE's IP packets to the MEC before they are encapsulated into GTP tunnels. All these solutions are not standard-compliant and require modifications on the eNB or/and the core network, so large deployment costs can be expected. Our solution does not make any changes on the existing LTE network elements.

3 MEC Deployment as a Middlebox

We adopt a middlebox approach to deploy the MEC platform in LTE networks. It sits on the S1 interface between the eNB and the S-GW, but in proximity of the former. Some traffic is routed to application servers on the MEC platform, and the other traverses the core network to the Internet by passing through the MEC. To make the MEC transparent to the existing network architecture, we shall address the following four issues without modifying any existing network components. First, how to intercept and forward the GTP packets which are routed directly between the eNB and the S-GW? Second, how to allow the MEC application servers to serve data packets which are embedded in GTP tunnels? Third, how to redirect MEC traffic to the MEC application servers while keeping the other traffic reaching the Internet? Forth, how to identify each GTP tunnel, which is dynamically built for an active UE, to serve each UE at run time?

3.1 Design Ideas

We address those four issues with the following four design ideas, respectively.

Proxy ARP. The MEC middlebox connects to the eNB and the S-GW with two NICs. By default, the MEC is unable to route GTP packets, since the destinations of those packets are either the eNB or the S-GW, but not the MEC. Normally, the eNB or the S-GW sends out ARP packets to resolve the peer's MAC address based on its IP. With the MEC, which splits the S1 into two network

segments, no response is made to those ARP packets. Therefore, the eNB or the S-GW does not know where to send packets at the link layer.

To this end, we employ the proxy ARP at the MEC to enable the eNB and S-GW to send the MEC GTP packets. It advertises the eNB of the MEC's MAC address to be associated with the S-GW's IP address, so the eNB considers the MEC as its next hop, instead of the S-GW. In the same way, the MEC can also become the next hop for the GTP packets sent from the S-GW to the eNB.

GTP Repackaging. The UE's IP packets are encapsulated in the GTP packets delivered over UDP/IP between the eNB and the S-GW. When receiving GTP packets, the MEC has to decapsulate the IP packets before redirecting them to its application servers. On the other hand, the IP packets sent from the servers to the UE have to be encapsulated back into the GTP tunnel.

This GTP repackaging method requires to differentiate multiple UEs so that each UE's packets can be encapsulated into the correct GTP tunnel, since the UEs have their own tunnels with different IDs. A pair of tunnel IDs, which identify two ends of a UE's tunnel, are included in the GTP packet header. The method thus needs to maintain a mapping table of each active UE's IP address and tunnel IDs. By checking the destination IP address of the packets, the MEC can formulate correct GTP headers to do encapsulation based on the table.

Traffic Redirection via DNS. The MEC has to redirect the UE's IP packets to its application servers. We enable this traffic redirection using the DNS service. Most applications rely on the DNS to resolve their servers' IP addresses before connecting to the servers, so the MEC can provide a DNS server that can return local IP addresses in response to the MEC applications' domain names. As a result, the UE's applications can directly communicate with their MEC servers using the MEC's local addresses. For the other non-MEC applications, the MEC's DNS server forwards their DNS queries to other name servers on the Internet.

Stateful Tracking of GTP Tunnels. Enabling the GTP repackaging requires the UE's IP address and tunnel IDs. We seek to prevent the MEC from querying the MME, which has each UE's information, so we extract the information from GTP packets using a stateful tracking. It is similar to the stateful firewall, but the MEC tracks the state of GTP tunnels instead of network connections. It learns the IP and tunnel IDs for each active GTP tunnel whenever a new GTP packet is observed. After a GTP tunnel's packets have been absent for a time period, its state is removed.

Note that whether to enable confidentiality protection on S1-U interface is based on the operator's decision, according to the 3GPP security standards [4]. In practice, the eNBs and S-GWs are usually placed in physically

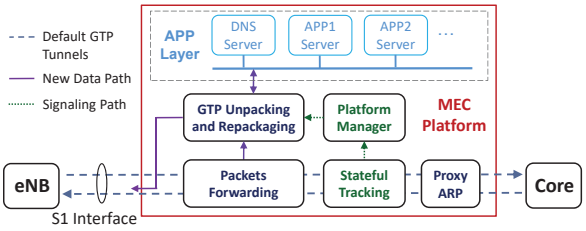


Figure 2: MEC platform architecture.

secured environments and their traffic does not traverse any insecure network components. So, the confidentiality protection is usually disabled by default. We have validated it in a commercial cellular platform containing Ericsson's small cell (i.e., eNB) and core network (i.e., Evolved Packet Core). We believe that operators hardly enable this security function due to the cellular network infrastructure as a closed system. However, if an operator enables this function, it needs to allow the MEC to acquire the security context of the S1-U interface. Since this security function is not standardized, how to support the MEC depends on each operator's specific design.

3.2 MEC Platform Architecture

We devise the architecture of an MEC middlebox platform based on the above design ideas, as shown in Figure 2. It sits between the eNB and the core (i.e., the S-GW) as a middlebox. There are three types of traffic paths. In addition to the paths of default GTP tunnels, along which packets are sent directly between the eNB and the core network, there are two new paths, data and signaling. The former is used for traffic redirection to consume MEC applications, whereas the latter is to collect necessary information that enables the MEC. The MEC platform contains five main modules and an application layer where a DNS server and application servers are deployed. We elaborate each of them below.

The proxy ARP module enables the system to answer the ARP requests sent from the eNB and the core, thereby being able to do packets forwarding between them. The packets forwarding module filters packets into three groups: the ones with the transport destination port 53, the ones with destination addresses as local IP addresses, and the others. It respectively forwards them to the DNS server, corresponding MEC application servers and the core network. Note that this forwarding module does not strip GTP headers off GTP packets, but only checks the values of several IP header fields.

The GTP unpacking/repackaging module takes care of the packets which need to be served by the MEC application servers. When they come from the forwarding module, this module needs to unpack GTP packets by getting rid of their GTP headers and then forwards IP packets to

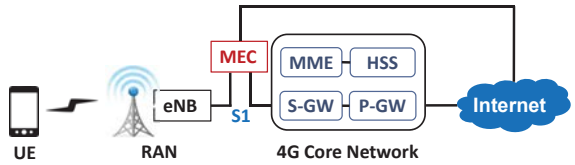


Figure 3: The prototype of the MEC middlebox in the 4G LTE network architecture.

Table 1: Platform information of the MEC middlebox.

Entity	Hardware	Software
4G Core	CPU: Intel Core i7-7700 3.60GHz RAM: 16GB	OS: Ubuntu 16.04.3 Kernel: 4.7.7-oaiepc
MEC	CPU: Intel Core i7-7700 3.60GHz RAM: 16GB	OS: Ubuntu 16.04.1 Kernel: 4.13.0-36-generic
eNB	CPU: Intel Core i7-7700 3.60GHz RAM: 16GB	OS: Ubuntu 14.04.1 Kernel: 3.19.0-61-lowlatency
Radio	eNB: Ettus USRP B210 UE: Huawei E3372h LTE dongle	N/A

the application layer. When they are sent from the application layer, it has to repackage them with GTP headers and then sends them to the eNB. To generate accurate GTP headers for the repackaging, it maintains a table to record the mapping of each UE's IP and tunnel IDs. The mapping entries are updated dynamically by the platform manager at run time. This information is collected by the stateful tracking module.

The application layer holds the servers of multiple MEC-enabled applications, which can be from third parties. They work as usual without the need of understanding GTP tunnels. It contains a DNS server that keeps a mapping of each application server's domain name and local IP address, and serves all the DNS requests.

4 Implementation and Evaluation

We prototype the MEC middlebox with the OAI cellular platform [1], which has the open source software for the LTE core network and eNB. Figure 3 shows the prototype architecture. We install the core network and the eNB on two PCs, and connect a radio board, Ettus USRP B210, to the eNB. The UE is a commodity laptop equipped with an LTE dongle, Huawei E3372h. The MEC is implemented on a PC with three network interfaces. Two of them connect to the core network and the eNB, whereas the other connects to the Internet, which is used when the MEC's DNS server requires to forward DNS requests to other name servers. Table 1 summarizes the detailed platform information.

4.1 Implementation

We elaborate on the implementation of each module.

Proxy ARP. We set one ARP rule on each of the two interfaces connecting to the eNB and the core network. We use the `arp` command with the `pub` option in Linux to configure the system to answer ARP requests. When receiving an ARP request with the S-GW's or the eNB's IP address from one interface, the system answers it with the interface's MAC address. It can thus enable the packets delivered between the S-GW and the eNB to be sent to the MEC at the link layer.

Packets Forwarding. We use the `iptables` command to configure the `PREROUTING` rules to forward all the incoming packets to the forwarding module. We develop a `Python` program to filter packets. It skips the GTP headers, but checks the destination IP addresses and ports in the IP headers. It classifies packets into three aforementioned groups and forwards them accordingly.

GTP Unpacking and Repackaging. We develop another `Python` program to unpack and repackage GTP packets based on a mapping table of IP addresses and tunnel IDs, which is maintained in the memory. It uses `RAW SOCKET` to forward the IP and GDP packets to the application layer and the eNB, respectively.

Stateful Tracking. We develop another `Python` program using the `tcpdump` command to sniff packets. Whenever any new tunnel IDs are observed in one GTP packet, it extracts its IP information and store this mapping in the memory, as well as forward it to the GTP repackaging module.

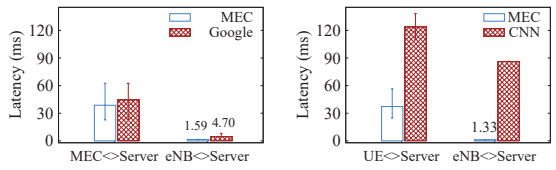
4.2 Evaluation

We evaluate the MEC prototype by examining the performance of web and video streaming services.

Web Service. We set up a web server on the MEC with a cloned CNN homepage. We examine two cases: the latency of the web access from the MEC and the actual CNN server. Each consists of two parts: the DNS query and the CNN web service. We test 100 runs for each case. Note that they rely on the DNS server on the MEC and the Google one on the Internet, respectively.

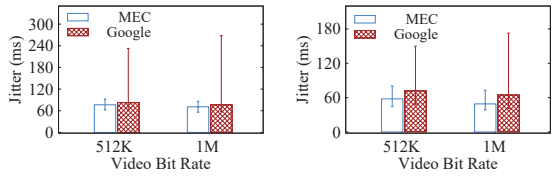
Figure 4 shows the 5th/50th/95th latency of the DNS and web services. We consider two latency types: overall latency (i.e., between the UE and the server) and the latency omitting the radio link (i.e., between the eNB and the server). For the former, the MEC platform can shorten the median latency of the DNS and web services by 13.36% and 69.86%, respectively. By ignoring the radio link latency, which can be much smaller in 5G, the latency gains can reach 66.18% and 98.46%, respectively.

Video Streaming Service. We employ the VLC media player [2] to stream a video from the MEC and the Google cloud to the UE. We consider two video qualities: 720p with 512K and 1M video bit rates. The other streaming settings are H264 video codec, MPEG audio codec, 128K audio bit rate, 24 frames per second, and



(a) DNS Query (b) CNN Web Service

Figure 4: 5th/50th/95th percentiles of latency for DNS and CNN web services on the MEC and the Internet.



(a) Jitters of Video Streams (b) Jitters of Audio Streams

Figure 5: 5th/50th/95th percentiles of jitters for video and audio streams with video bit rates 512K and 1M.

RTP/UDP transport protocols. We test 5 runs for each video quality and each run takes 5 minutes.

We examine the performance based on video frame loss and the jitters of both video and audio streams. In the MEC case, the average frame loss rates for those two bit rates are 0.13% and 0.27%, respectively. In the Google case, they are 3.55% and 3.43%, respectively. The MEC gains more than 3% loss reduction. The 5th/50th/95th jitters of the video and audio streams for these two cases are shown in Figure 5. For the video streams, the MEC case shortens the 95th jitter of the 512K and 1M bit rates from 232.7 ms to 92.3 ms (60.34% reduction) and from 268.1 ms to 86.2 ms (67.85% reduction), respectively. For the audio streams, it outperforms the Google case with 46.40% and 57.71% jitters reduction, respectively.

5 Discussion

We discuss several remaining issues in this section.

Scalability Challenges. Our current design can be scaled up to support multiple eNBs with only one MEC platform. There are two main challenges. First, the computing power demand of processing GTP packets can greatly increase with the number of serving eNBs. Second, context management can be more complex, since the eNBs may have different localized applications and network context.

MEC Applications. It is anticipated to have many applications installed on the MEC. The cloud computing technology (e.g., OpenStack) can enable the MEC to easily host multiple applications, and the virtualization (e.g., containers or virtual machines) can allow the applications to be migrated or to do state transfer.

Mobility Management. When the UE handovers between eNBs, coordination between MEC middleboxes is needed to migrate application states and user context. We can design a distributed coordination function to avoid requirements from the network infrastructure. We plan to address it in our future work.

Security and Billing. The security and billing issues can come from two sources. First, though the MEC middlebox can be deployed by carriers, the applications running on it may be from third parties. To defend against malicious applications, the MEC can rely on virtualization technologies to isolate applications and monitor them at run time. Second, since the UE's traffic towards the MEC does not traverse the core network, malicious UEs may access MEC applications without authorization or being billed. The MEC should provide access control against the UE's traffic so that it cannot reach its unauthorized applications. It should also generate eligible charging data records, where the carrier can specify charging rules for applications, to the core network.

Fault Tolerance. The MEC platform may fail, so we can prepare a backup MEC middlebox for fault tolerance. To minimize the impact, we can employ one of current virtualization backup technologies to restore application states on the backup MEC.

6 Conclusion

The MEC has been considered as a key enabler of low-latency performance in cellular networks. In this work, we design a middlebox approach for the MEC deployment. By sitting between the eNB and the S-GW, it filters, manipulates and forwards packets to enable MEC applications. It is easy to install and has low deployment cost that current network elements need not be modified. We validate its viability through an MEC prototype with the OAI cellular platform, and our preliminary result shows the effectiveness of latency reduction benefits. This middlebox approach not only fits carriers or small and medium enterprises (SMEs) for immediately enabling the MEC technology in existing 4G networks, but also can be a reference design for the 5G MEC.

7 Acknowledgments

We thank our shepherd Weisong Shi for his help, and also thank the anonymous reviewers for their valuable comments on improving this paper. This work was partially supported by the Ministry of Science and Technology, Taiwan, under grant numbers 106-2622-8-009-017 and 106-2218-E-009-018, and by the H2020 collaborative Europe/Taiwan research project 5G-CORAL (grant number 761586).

References

[1] OAI (OpenAirInterface): A Platform with Open Source Software for the Core Network, Access Network and User Equipment of 3GPP Cellular Networks., 2018. [Online] Available: https://www.openairinterface.org.

[2] VLC: A free and open source cross-platform multimedia player and framework that plays most multimedia files and various streaming protocols., 2018. [Online] Available: https://www.videolan.org.

[3] 3GPP. TS29.060: General Packet Radio Service (GPRS); GPRS Tunneling Protocol (GTP) across the Gn and Gp interface, Apr. 2010.

[4] 3GPP. TS33.401: 3GPP SAE; Security architecture, Sep. 2013.

[5] 3GPP. Technical Specification Group Services and System Aspects; System Architecture for the 5G Systems: Stage 2 (Release 15). *3GPP Standard TS 23.501 V0.4.0* (2017).

[6] 3GPP. TS23.002: Network architecture, Mar. 2017.

[7] 3GPP. TS29.281: General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U), Dec. 2017.

[8] 3GPP. TS36.413: LTE; Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 Application Protocol (S1AP), Jan. 2018.

[9] AHMED, A., AND AHMED, E. A Survey on Mobile Edge Computing. In *International Conference on Intelligent Systems and Control (ISCO)* (2016), IEEE.

[10] CHANG, C.-Y., ALEXANDRIS, K., NIKAEIN, N., KATSALIS, K., AND SPYROPOULOS, T. MEC Architectural Implications for LTE/LTE-A Networks. In *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture (MobiArch)* (2016), ACM.

[11] ETSI. Mobile Edge Computing; A Key Technology towards 5G. *ETSI White Paper No. 11* (2015).

[12] ETSI. Mobile Edge Computing (MEC); Framework and Reference Architecture. *ETSI GS MEC 003* (2016).

[13] ETSI. MEC Deployment in 4G and Evolution Towards 5G. *ETSI White Paper No. 24* (2018).

[14] HUANG, A., NIKAEIN, N., STENBOCK, T., KSENTINI, A., AND BONNET, C. Low Latency MEC Framework for SDN-based LTE/LTE-A Networks. In *International Conference on Communications (ICC)* (2017), IEEE.

[15] HUANG, S.-C., CHEN, B.-L., LUO, Y.-C., CHUNG, Y.-C., AND CHOU, J. Application-aware Traffic Redirection: A Mobile Edge Computing Implementation toward Future 5G Networks. In *International Symposium on Cloud and Service Computing (SC2)* (2017), IEEE.

[16] MACH, P., AND BECVAR, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Communications Surveys and Tutorials 19*, 3 (2017), 1628–1656.

[17] MAO, Y., YOU, C., ZHANG, J., HUANG, K., AND LETAIEF, K. B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys and Tutorials 19*, 4 (2017), 2322–2358.

[18] MORRIS, I. ETSI Drops Mobile From MEC. *Light Reading* (2016).

[19] TALEB, T., SAMDANIS, K., AND MADA, B. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Communications Surveys and Tutorials 19*, 3 (2017), 1657–1681.