On-site Evaluation of a Software Cellular based MEC system with Downlink Slicing Technology

Koichiro Amemiya*†, Yuko Akiyama‡, Kazunari Kobayashi‡, Yoshio Inoue‡, Shu Yamamoto† and Akihiro Nakao†
*Fujitsu Laboratories Ltd., Kawasaki, Japan, Email: amemiya.kouichi@jp.fujitsu.com
†The University of Tokyo, Tokyo, Japan, Email: shu@iii-u-tokyo.ac.jp, nakao@nakao-lab.org
‡Fujitsu Limited, Kawasaki, Japan, Email: {akiyama.yuko,kazu_kobayashi,yoshio.inoue}@fujitsu.com

***Abstract*—MEC (Mobile/Multi-access Edge Computing) has recently caught much attention for processing data in the vicinity of user equipment (UE) to reduce latency for real-time applications. One of the challenges of MEC is to reduce the cost of deploying computational resources near the cellular base stations. Recently, software implementation of cellular base stations is emerging as it allows colocation of access point functionalities and those of MEC within the same commodity hardware, e.g., using containers, thus facilitates deployment of MEC without incurring much cost. In this paper, we posit that one of the most significant challenges for realizing softwarized base stations with MEC capability is to enable resource isolation among slices, especially isolating low latency slice as the primary concern of MEC is to enable low-latency application. Our contributions are threefold. First, we define the architecture of MEC infrastructure in softwarized cellular network. Second, we measure the actual latency and throughput of on-site MEC in a softwarized cellular network. And at last, we propose a novel slicing method for softwarized base stations to isolate a low latency slice from a broadband one. Our evaluation shows that the proposed method enables reasonable resource isolation, achieving the same minimal latency even with a competing broadband slice as that without any other slice.**

***Keywords*—Edge Computing, MEC, slicing, 5G, IoT**

I. Introduction

Huge data from IoT devices is processed into information valuable for society and utilized by various applications and services. To collect and process the valuable data for the IoT applications in optimal time and place, low latency and broad bandwidth are the critical requirements for the network infrastructure. For example, cooperative driving [8] requires low latency mobile network. Local dynamic mapping for safe driving, identifying the other vehicles, pedestrians or obstacles on the road with the onboard camera and image processing, requires broad uplink bandwidth as well as low latency. And content delivery needs even broader downlink bandwidth for the mobile network.

Multi-access Edge Computing (MEC) [2] with a high-performance cellular network, such as the fifth-generation mobile network (5G) [4] , is supposed to address these requirements by processing data in the vicinity of user equipment (UE). MEC provides a computing platform with multiple access media, such as mobile network, optical fiber, and wireless LAN and deployed besides users. And network system with MEC provides high responsiveness and capacity that accommodates heavy traffic from the wireless or wired access network. Service applications can extract valuable information from the stored raw data at the edge of the network and distribute the information quickly to the local devices utilizing the computing and storage resources of MEC without wasting the network resources.

Network carriers and MEC platform service providers are expected to deploy the MEC system all over the country for bringing IoT applications to nation-wide availability. However, network-wide deployment of the MEC platform with 5G and high-performance cellular network equipment costs much. Most of the current 4G cellular network equipment utilizes the specially manufactured expensive hardware [12]. We posit that this cost for wide deployment of MEC is the most prohibiting obstacles for infrastructure and service providers.

In the light of this observation, we believe that softwarization of MEC and cellular network system is the rescue to reduce the CAPEX of deployment cost of MEC. Thanks to the progress of the microprocessors not only the network and the application functionalities but also the heavy processing for radio access network (RAN) become available on the commonly used inexpensive server hardware, e.g., IA server [18]. If mobile network carriers migrate the cellular network equipment to the softwarized one using the commodity hardware they can install the MEC functionalities into the same infrastructure with cellular network functionalities saving the additional costs for the place and the hardware for MEC.

However, the slicing method especially keeping the performance of the low latency applications even when throughput intensive slices exist within the same infrastructure has not been established. We believe the resource isolation in softwarized cellular systems need to be carefully designed and implemented. FLARE [7] and Orion [10] propose methods to isolate the radio network and the computing resources among slices. But when heavy downlink traffic is generated at the MEC, traffic congestion occurs at the ingress of the EPC and the eNodeB and the congestion could ruin the low latency. CellSlice [11] proposes scheduling method to isolate the incoming traffic of the eNodeB. But the proposed scheduling algorithm does not secure the low latency though

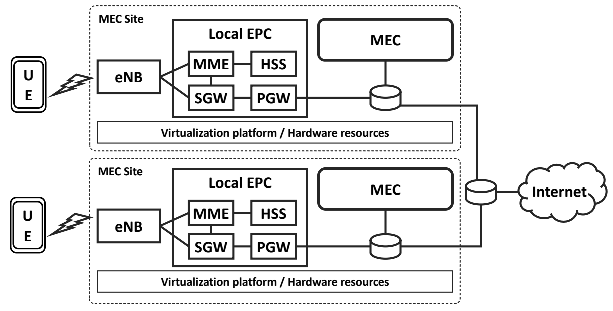
MEC Site
Local EPC
MEC
UE
eNB
MME
HSS
SGW
PGW
Virtualization platform / Hardware resources
MEC Site
Local EPC
MEC
Internet
UE
eNB
MME
HSS
SGW
PGW
Virtualization platform / Hardware resources

Fig. 1. MEC deployment architecture

it isolates the bandwidth of the incoming traffic.

To address the issue of resource isolation in softwarized cellular systems, we propose a MEC slicing architecture where resource control functionalities are distributed among eNodeB, EPC, and MEC. Distributed resource controllers cooperate to achieve end-to-end slices. And we propose a downlink slicing method where the latency-aware scheduling mechanism within the softwarized EPC works in accordance with the RB scheduler within eNodeB to achieve low latency slice. Additionally, we construct on-site softwarized MEC system and measure the actual performance.

Our primary contributions of this paper are threefold:

- We define the architecture of MEC infrastructure in softwarized cellular network.
- We measure the actual latency and throughput of on-site MEC in softwarized cellular network.
- We propose a novel slicing method for softwarized cellular network to isolate a low latency slice from a broadband one.

Our on-site evaluation shows that the proposed method enables reasonable resource isolation, achieving the same minimal latency even with a competing broadband slice as that no other slice exists.

The rest of the paper is organized as follows. Section II defines the softwarized MEC architecture. Section III presents our proposal about the slicing method for the MEC with a softwarized cellular network. We show the specifications of our on-site testbed in Section IV and show the results of performance tests in Section V and evaluate the proposed slicing method in Section VI. We discuss the related works in Section VII and conclude the paper in Section VIII.

II. The architecture of the softwarized MEC

In this section, we show a basic architecture of the MEC system.

A. MEC deployment

MEC whitepaper [1] describes four MEC deployment patterns; (a) Bump in the wire, (b) Distributed EPC, (c) Distributed S/PGW, (d) Distributed SGW with Local Breakout and we adopt (b) as our basic architecture (Fig. 1) for stable performance.

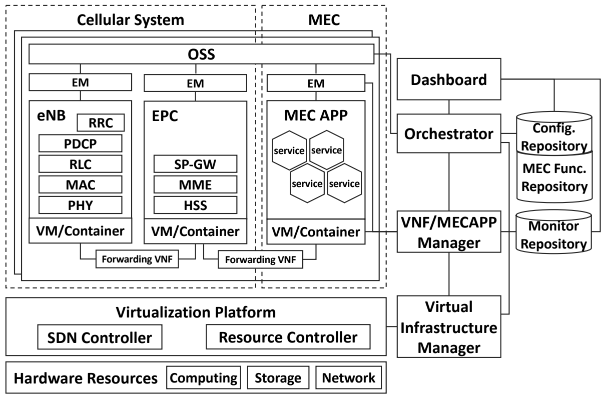

Fig. 2. Softwarized MEC architecture

In current LTE system, eNodeB accommodates wireless access of UEs, and each eNodeB connects to centralized PGW through SGW. Each UE is assigned IP address after going through the PGW. The UE can access the services using the assigned IP address. But the traffic from UEs through eNodeB and SGW are met at the EPC output, and the concentrated traffic causes congestion. The congestion could degrade the service level.

To avoid the congestion, the traffic of UEs should be processed distributedly. In the MEC system of distributed EPC UEs are assigned their IP address besides distributed eNodeB and EPC. The services on distributed MEC servers can receive the traffic from the limited numbers of the UEs. The latency and throughput for each UE become stable when the number of UEs is limited.

B. Softwarized MEC architecture

Fig. 2 shows softwarized MEC architecture. We design the architecture based on the NFV architecture with network slicing support [5]. With this architecture, the mobile cellular components, as well as MEC service components, can be deployed within the virtualization environment sharing the same hardware resources. The virtualization platform isolates the virtual resources among MEC functionalities and slices.

The orchestrator organizes the virtual resources and functionalities through the virtualization infrastructure manager and virtual functionality manager (VMF, MECAPP manager) respectively. The softwarized mobile cellular functions can enjoy the management functionalities of virtualization platform such as auto scale, instant deployment, and continuous integration.

III. Slicing of the softwarized MEC

In MEC system, there are multiple resources could be the bottleneck for achieving the requirements of services about performance. To guarantee the requirements, the location of the slicing control functionalities should be arranged to compensate the performance degradation at each resource.

In the softwarized cellular network, the computing resources are important for the better performance. The processing in

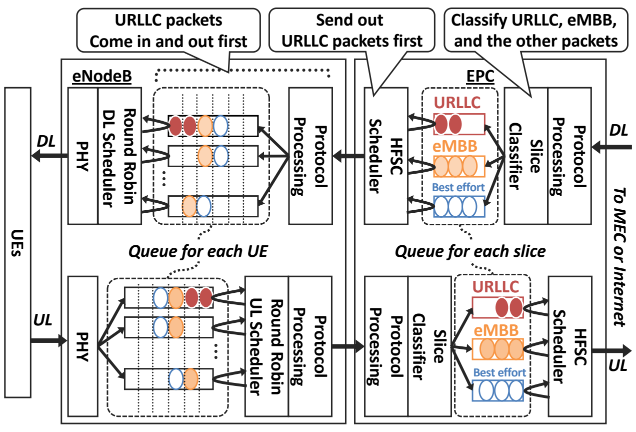

Fig. 3. Downlink Slicing offload Method

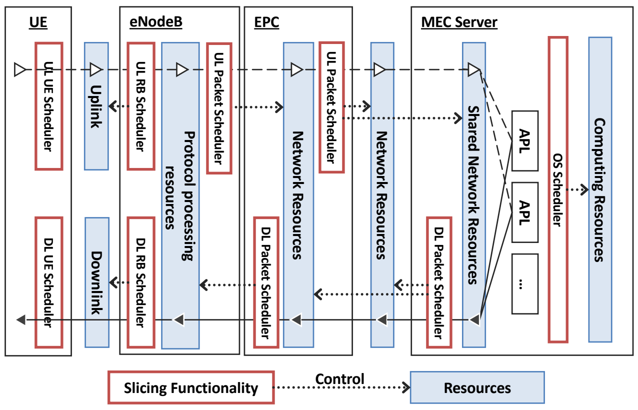

Fig. 4. MEC Slicing Architecture

eNodeB, especially PHY and MAC layer processing, use much amount of computing resources. The processing for PHY and MAC that have been implemented on hardware devices such as DSP or FPGA before softwarization [12], is important to provide higher performance on the software-based platform.

The slicing functionality is usually implemented within the uplink or downlink schedulers in MAC layer. These schedulers have to decide the allocation of the wireless resource blocks to the communication packets every 1 millisecond. To achieve the requirements for the wireless communication performance, such as maximization of the number of the accommodated users, minimization of the latency or fairness among, various optimization algorithms have been proposed [16] [15]. With the help of the DSP or FPGA, it is possible to allocate the resource blocks to UEs' communication packets within 1 millisecond such that suffices these complex requirements. But as for softwarized platform to preserve the performance of the PHY layer, it is probable that there are not so much computing resources saving for the RB scheduling process depends on the deployment environment.

We propose a downlink slicing method. The scheduler within eNodeB, the packet scheduler within EPC and MEC work consistently and share the burden of the processing to achieving slicing. The packet schedulers on EPC and MEC identify the slices that each packet belongs to. And they classify the packets into the queue based on the identified slice information. Then, they allocate the network resources between MEC and EPC or EPC and eNodeB to the packets according to the specification of the slices such as real-time, link sharing or dedicated bandwidth.

As a packet scheduler for resource isolation, token bucket based flow control has been used. Predefined numbers of Tokens are allocated to each service (or slice) at each time slot. Each service can use network or computing resources corresponding to the amount of the allocated tokens within a time slot. Additionally, unused tokens within the time slot can be used at the following time slot. The token bucket mechanism can control average usage of resources and works well for resource isolation between throughput intensive slices. But it can not control the latency of the services. When one slice requires low latency (e.g. URLLC) and the other slice requires higher throughput (e.g. eMBB) on a shared physical infrastructure, the latency of the service in low latency slice could be degraded by the interference from the throughput intensive slice with heavy traffic.

We adopt Hierarchical Fair Service Curve (HFSC) [17] to isolate the low latency slices using computing resources within EPC (Fig. 3). HFSC is composed of the two-step decisions. The first step decides whether the received packet should be processed in a real-time manner or a link sharing manner. For the purpose, HFSC manages eligible curve for each hierarchical class. The curve indicates a potential conflict that violates at least one packet's deadline. As a next step for the packet which is decided to be processed in a real-time manner, HFSC selects which packet should be processed first. In this step, HFSC uses deadline curve. These curves are updated every time when the packet of the session is received considering the current status.

Based on the discussion above we define the MEC slicing architecture among eNodeB and EPC in Fig. 3. We use a simple round-robin algorithm for UL and DL RB scheduler. And the downlink packet scheduler at the downlink egress of EPC uses HFSC algorithm for controlling the latency of the downlink traffic. We achieve low latency among eNodeB and EPC utilizing these schedulers saving the computing resources within eNodeB.

Fig. 4 shows the architecture as a whole. UL packet scheduler in eNodeB and DL packet scheduler in MEC are responsible for controlling network resources on EPC. UL packet scheduler in EPC and DL packet scheduler in MEC are responsible for shared network resources in MEC servers. These schedulers use HFSC algorithm to achieve low latency. Additionally, the UL and DL scheduler with HFSC algorithm in UE also contribute to achieve low latency slices. Finally, a scheduler in the MEC operating system controls the computing resources on MEC, such as CPU or memory.

IV. Software Cellular Network with MEC Testbed

To evaluate the performance of the MEC with softwarized cellular network we constructed on-site MEC platform in our campus [13]. The specification of the software LTE base station and EPC is as follows.

- Based on an opensource software OpenAirInterface (OAI) [18]
- All Protocol stack from L1-L3 within eNodeB, except for the wireless analog signal processing, is implemented with software
- eNodeB and EPC functionality are implemented on the commonly used commercial hardware. UEs and radio devices are also the commonly used products
- eNodeB, EPC and MEC functionality is deployed using virtualization platform for the flexible and quick deployment and operation.

The radio interface of the testbed uses 1.7GHz central frequency and 5MHz bandwidth. The modulation method is FDD. The detailed specifications are shown in TABLE I.

Fig. 5 shows the architecture of our MEC testbed system. We use independent virtualization platforms for cellular network functionalities and MEC application respectively. We adopt KVM as a virtualization platform for cellular network functionalities since they require firm resource isolation. And we use docker [20] container for MEC applications as they are supposed to be developed and repeatedly modified with continuous integration. We use Kubernetes [19] to manage the MEC applications among MEC application server cluster. With the help of the docker, Linux cgroup, and Kubernetes, we can execute various service applications within the isolated computing and network resources on the MEC application servers.

V. Performance test of the MEC Testbed

In this section, we evaluate the performance of the on-site softwarized cellular testbed equipped with MEC. We measure the latency and the throughput of the system because they are important performance metrics for typical MEC applications such as IoT or vehicle services. And for the evaluation, we use ICMP ping, HTTP client/server, and MQTT subscriber/publisher for the latency measurement and iperf client/server for the throughput measurement. We measure the same metrics for the cloud system to evaluate the effectiveness of the MEC system.

TABLE I
SOFTWARE RADIO SPECIFICATIONS

Central Frequency	Downlink: 1842.4 5MHz Uplink: 1747.4 MHz
Bandwidth	5.0 MHz
Multiplex Method	Downlink: OFDMA Uplink: SC-FDMA
Modulation	Downlink: QPSK, 16QAM, 64QAM Uplink: QPSK, 16QAM
Transmission Power	mobile base station: 2mW, 200mW

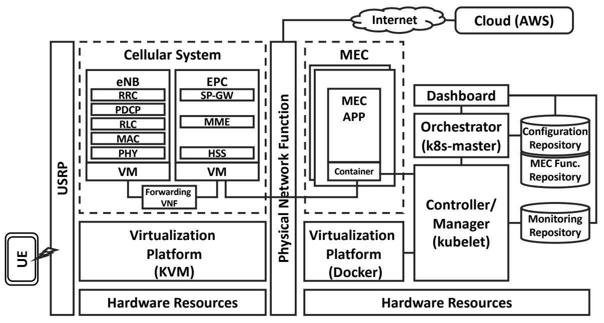

Fig. 5. MEC testbed

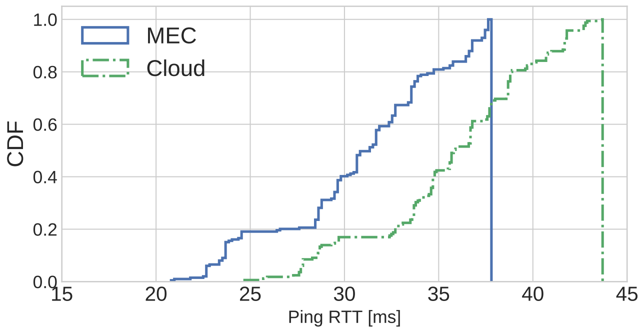

Fig. 6. Ping RTT

A. Latency

At first, we measure the basic performance of network latency with ICMP ping. The result is shown in Fig. 6. Ping RTT median between UE and MEC was 31milliseconds(ms) and 36 ms for UE and Cloud. Additionally, ping RTT between eNodeB and EPC is 60 microseconds, RTT between EPC and MEC is 200 microseconds. We can conclude that a large amount of delay occurs at the wireless network between eNodeB and UE from these results. As for the comparison with the cloud environment, the MEC system provides shorter RTT by 5-6 ms than the RTT of the cloud.

Next, we measure the RTT of HTTP. HTTP is the most popular protocol for the internet. And it is possible for HTTP to be used in IoT services such as data upload or contents delivery. For the measurement, HTTP servers are deployed on MEC and the cloud respectively and UE sent HTTP requests to the HTTP servers. UE is configured to send HTTP requests with keep-alive ON. Fig. 7 shows the result. The RTT median for MEC is 31 ms and 14ms at the minimum. The RTT median for the cloud is 39ms and 21ms at the minimum. The difference between the results of MEC and the cloud is reasonable since there is 5-6ms delay between EPC and the cloud from the measurement results about ping.

Sometimes the RTT became larger up to dozens of milliseconds. Since HTTP is connection based protocol and sometimes the connections between UE and MEC/cloud HTTP servers is lost and reconnect of TCP connection happens. And it is probable that the failure of uplink scheduling between UE and eNodeB causes the reconnection.

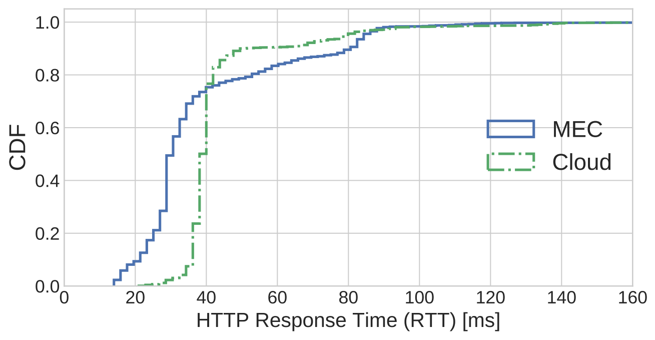

Fig. 7. HTTP RTT

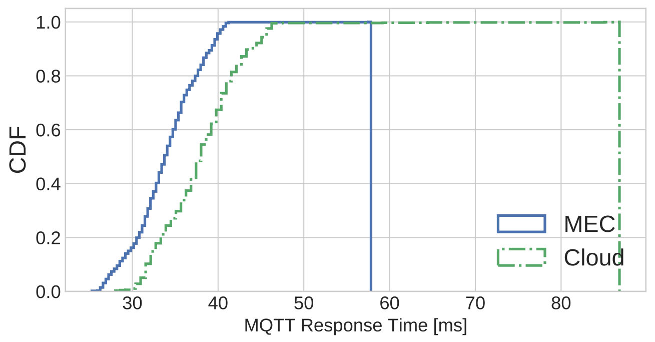

Fig. 8. MQTT RTT

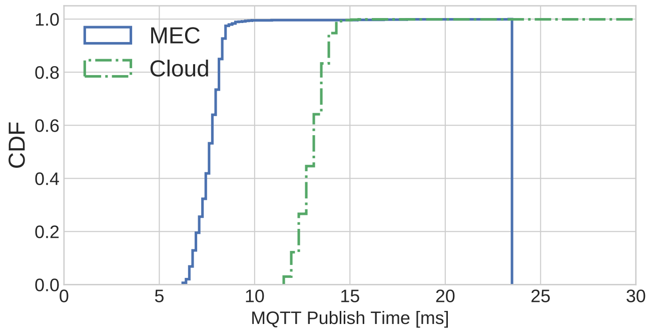

Fig. 9. MQTT Unidirectional (Downlink)

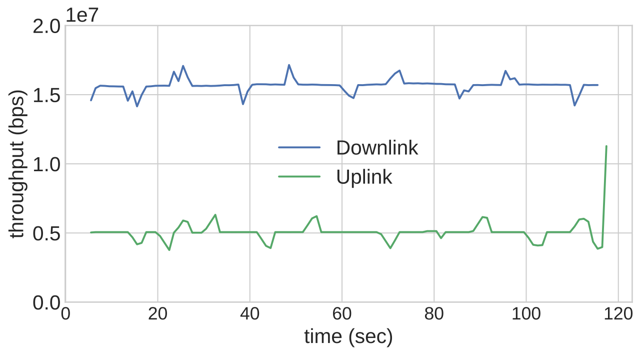

Fig. 10. iperf throughput

And we measure the delay for MQTT subscribe/publish. It is also valuable to evaluate the latency of the MEC with MQTT protocol since MQTT is the most popular protocol in IoT services and required low latency to notify data to field devices quickly. A UE subscribes to the MQTT broker on the MEC server and the cloud server. Then the UE sends publish messages to each MQTT server and receive the notification messages which are triggered by its own publish. Fig. 8 shows the MQTT RTT. The RTT is calculated from the difference between the time when the UE sends the publish message and the time when the UE receive the notification message from the broker. The MQTT RTT median for MEC is 34 ms, and 26 ms at the minimum. And for the cloud, the RTT median is 42 ms, and 31 ms at the minimum.

From the RTT CDFs for various protocols, there are the points where the gradient of the CDF changes. These are caused by the uplink and downlink scheduler of the eNodeB. For the uplink, UE sends scheduling request (SR) with the intervals of 10 ms. If there are other UEs sending SR, eNodeB uplink scheduler allocate the RBs in a round-robin manner. If the UE failed to get the RB at the time, the allocation of the RBs for the UE is postponed to the next timing. In the next opportunity, the UE which failed to get RB is the priority. It could be happened for the sending packet to be delayed 0 to 20 ms. Similar mechanisms are working at the downlink scheduler and delay could occur there too. And over 4 ms of delay could occur at the RB scheduler. The scheduler works every 1 ms and allocates the RB in a subframe of 4 milliseconds after with our implementation. Additionally, the queued packet at the scheduler could receive a few more milliseconds delay. Some packets luckily succeed to get the RBs at the first trial, others fail and get the RBs at the next scheduling opportunity after 1ms. These variations cause the variation of gradient on the CDF.

We also evaluate the latency for the uni-directional MQTT notification from MEC server or cloud server to UE, Fig. 9. For MEC the latency is 6 ms at the minimum, and the median is 8 ms. And the protocol processing within eNodeb takes a few milliseconds. The notification latency from cloud server is 11 ms at the minimum, and the median is 13 ms. The notification latency from MEC is appropriate considering the scheduling mechanisms described above and that the protocol processing within eNodeB takes a few milliseconds. The notification latency from the cloud is also appropriate for the measured delay between EPC and the cloud.

B. Throughput

Fig. 10 show the throughput of the on-site MEC testbed. We get the throughput of 16Mbps for downlink and 5Mbps for uplink. These performance results are consistent with the specification shown in TABLE I.

VI. Slicing method Evaluation

We evaluated the slicing functionality for the downlink of our testbed. We set up low latency slice and broadband slice on the MEC testbed and deployed MQTT broker within the low latency slice and iperf server within the broadband slice as shown in Fig. 11. Downlink packets sent from MEC are

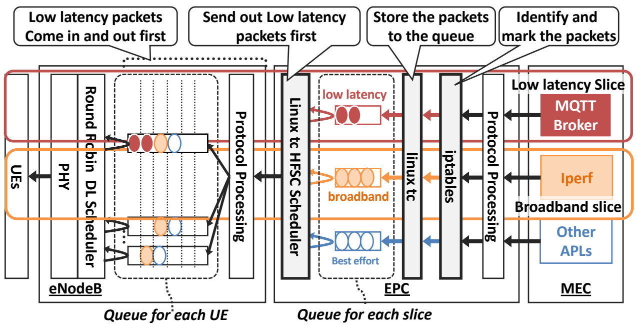

Fig. 11. slicing evaluation setup

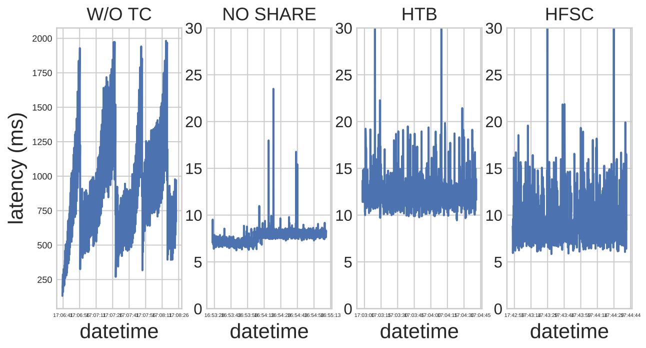

Fig. 12. Latency of low latency slice

sorted out and added marks to identify the slice to which the packets belong within EPC. And the packet scheduler in EPC controls the timing of sending packets according to the mark on each packet. Downlink scheduler on eNodeB allocates resource blocks (RBs) of the wireless network to the incoming packets and send it to the UE. The packet scheduler is Linux TC basis, and we evaluated two types of scheduling algorithms, HTB and HFSC. Downlink scheduler in eNodeB uses round robin algorithm among connected UEs to assign RBs.

In this slicing evaluation, MQTT and iperf traffic are sharing the RAN and MEC resources. If the isolation between slices is incomplete, the latency within the low latency slice will be degraded, or the iperf traffic within the broadband slice cannot utilize the remaining bandwidth efficiently. The time series variation of the latency and the throughput are shown in Fig. 12 and Fig. 14.

Fig. 12 shows that the latency of the low latency slice (MQTT packet) is 131 milliseconds(ms) at the minimum without traffic control (W/O TC) and 6 milliseconds at the minimum when the low latency slice uses the infrastructure exclusively(NO SHARE). HFSC improves the latency comparing with HTB packet scheduling algorithm though the variation of the latency is larger than that of no resource sharing. Fig. 13 shows the CDF of the latency for no resource sharing, sharing with HTB and sharing with HFSC. The minimum latency for the resource sharing with HFSC is the same as that of no resource sharing, 6 ms. But the standard deviation for HFSC is 3 ms, and 1 ms for no resource sharing. This result means that the HFSC scheduling within EPC only could not eliminate the influence of the sibling throughput intensive slice completely. There are the protocol processing and the processing for allocating the radio access network resources within eNodeB after the proposed scheduler within EPC for downlink. To achieve more precise slicing we need to introduce isolation mechanisms for the protocol processing and customize the RB scheduler within eNodeB.

Fig. 14 shows that with HFSC the throughput intensive slice succeed to get maximum throughput that is same as the throughput without resource sharing shown in Fig. 10. But with HTB low latency slice keeps some part of bandwidth and the throughput intensive slice does not achieve the maximum

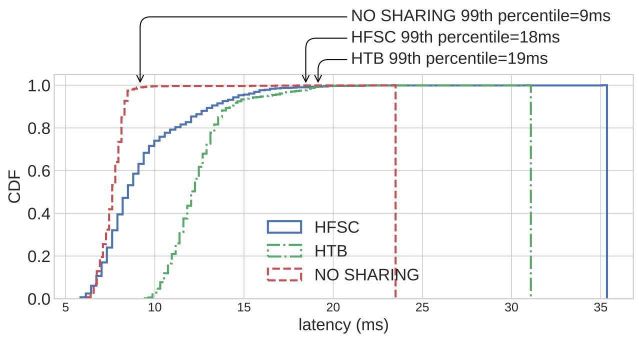

Fig. 13. latency CDF of low latency slice

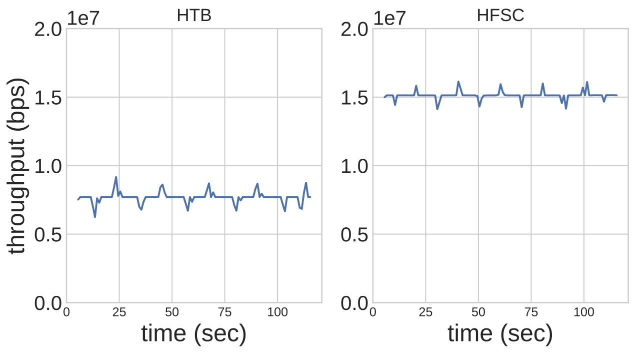

Fig. 14. Downlink Throughput of broadband slice

throughput, 16Mbps. To achieve the slicing of the MEC system the scheduling algorithms for slicing has to be equipped with the control scheme where the packets of real-time slices and that of bandwidth consuming slices are identified and processed accordingly like HFSC.

VII. Related Work

FLARE [7] shows a slicing method by deploying different instances of docker container for each slice allocating wireless bandwidth to each slice statically. Orion [10] achieves RAN slicing more flexible manner. Orion defines vRRB (virtual Radio Resource Block) which bundles one or more PRBs (Physical Resource Block) in one Subframe. Slicing is achieved by controlling the visibility of vRRBs from each Slice. With this mechanism, dynamic allocation of frequencies to slices is possible. Slice characteristics that can be realized by Orion are (i) fixed assignment of RB, (ii) average RB number in 100 ms window unit and (iii) average throughput per 100 ms window. FLARE and Orion do not consider the latency of the slices though they achieve the isolation of the bandwidth among slices. And when heavy downlink traffic is generated at the MEC, traffic congestion occurs at the ingress of the EPC and the eNodeB. The congestion cannot be solved only by the mechanisms within eNodeB and EPC like FLARE or Orion.

CellSlice [11] shows a slicing method similar to ours. They deploy a gateway called as CellSlice between EPC and eNodeB and control slicing there. Hierarchical packet scheduler within the CellSlice gateway sorts out incoming packets and controls the timing of sending packets to achieve slicing. But they do not introduce a latency related control method of the packets, though they care about the bandwidth of each slice.

VIII. Conclusion

Our contributions are threefold. First, we define the architecture of MEC infrastructure in softwarized cellular network. The architecture enables softwarized MEC servers and cellular network components to be colocated within the same infrastructure and managed consistently and facilitates deployment of MEC without incurring much cost. Second, we evaluate the performance of an on-site MEC in softwarized cellular network. The on-site system provides MEC applications 14 ms RTT, and 6 ms latency with 1 ms jitter for the downlink notification at the minimum. According to [6], the proposed system has enough capability to provide intelligent transport system which requires 10 ms end-to-end latency with 20 ms jitter. And third, we propose a novel slicing method for softwarized base stations to isolate a low latency slice from a broadband slice. We implement an HFSC based latency aware packet scheduling mechanism within the softwarized EPC to solve the congestion and prioritize the packets of low latency slices at the downlink ingress of the eNodeB. The proposed method provides a latency of 6 ms at the minimum with 3 ms jitter for the low latency slice even when a competing broadband slice exists within the same infrastructure. The minimum latency is the same as the value when no slice other than the low latency slice exists. Intelligent transport system requires latency under 10 ms sharing infrastructure among the traffic with various size of payload [6]. And it is possible that different applications share the same infrastructure. This result shows that our proposed method achieves low latency slice practical and cost-effective manner.

For future work, we plan to isolate the protocol processing functionalities before allocation of radio access network resources within eNodeB to achieve more precise isolation. And it should be noted that softwarized system does not provide comparable capacity to the hardware-based system because of the limited computing resources. We plan to study optimal system design including the RB scheduler, supporting scheduler outside of eNodeB, and load sharing scheme among distributed softwarized cellular system with MEC to achieve both latency and capacity requirements for some specific applications.

References

[1] ETSI White Paper No. 24, “MEC Deplyments in 4G and Evolution Towards 5G,” Feb. 2018.

[2] C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile Edge Computing A key technology towards 5G,” ETSI White Paper No. 11 Mobile, no. 11, pp. 116, 2015

[3] ONF TR-526, “Applying SDN Architecture to 5G Slicing,” Apr. 2016.

[4] ITU-R M.2083-0, “IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond,”, Sep. 2015.

[5] ETSI GR NFV-EVE 012, “Network Functions Virtualisation (NFV) Release 3 ; Evolution and Ecosystem ; Report on Network Slicing Support with ETSI NFV Architecture Framework,” V3.1.1, 2017.

[6] 3GPP TS 22.261, “Service requirements for the 5G system,” V16.4.0, Jun. 2018.

[7] A. Nakao, P. Du, Y. Kiriha, F. Granelli, A. A. Gebremariam, T. Taleb, and M. Bagaa, “End-to-end Network Slicing for 5G Mobile Networks,” J. Inf. Process., vol. 25, no. 0, pp. 153163, 2017.

[8] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, “Layered vehicle control system coordinated between multiple edge servers,” 2017 IEEE Conf. Netw. Softwarization Softwarization Sustain. a Hyper-Connected World en Route to 5G, NetSoft 2017, vol. 2017July, pp. 15, 2017.

[9] X. Foukas, N. Nikaein, M. M. Kassem, K. Kontovasilis, and M. K. Marina, “FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks,” Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol. ACM, pp. 427441, 2016.

[10] X. Foukas, M. K. Marina, and K. Kontovasilis, “Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture,” Proc. 23rd Annu. Int. Conf. Mob. Comput. Netw. - MobiCom 17, pp. 127140, 2017.

[11] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, “CellSlice: Cellular wireless resource slicing for active RAN sharing,” 2013 5th Int. Conf. Commun. Syst. Networks, COMSNETS 2013, 2013.

[12] H. Watanabe, S. Hirasawa, K. Suzuki, and R. Karino, “Evolved Node B on LTE system for NTT DOCOMO,” Fujitsu Sci. Tech. J., vol. 48, no. 1, pp. 2126, 2012.

[13] Y. Akiyama, K. Kobayashi, S. Yamamoto, A. Nakao, “Field Trial of Software based Cellular System,” Proceedings of the 2018 IEICE General Conference, B-6-61, 2018.

[14] J. Ordonez-Lucena, P. Ameigeiras, Di. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges,” IEEE Commun. Mag., vol. 55, no. 5, pp. 8087, 2017.

[15] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, Downlink packet scheduling in LTE cellular networks: Key design issues and a survey, IEEE Commun. Surv. Tutorials, vol. 15, no. 2, pp. 678700, 2013.

[16] N. Abu-Ali, A. E. M. Taha, M. Salah, and H. Hassanein, Uplink scheduling in LTE and LTE-advanced: Tutorial, survey and evaluation framework, IEEE Commun. Surv. Tutorials, vol. 16, no. 3, pp. 12391265, 2014.

[17] I. Stoica, H. Zhang, and T. S. Eugene Ng, “A hierarchical fair service curve algorithm for link-sharing, real-time and priority services,” In Proceedings of the ACM SIGCOMM 1997 (SIGCOMM 97), pp. 249-262, 1997.

[18] OpenAirInterface: http://www.openairinterface.org/

[19] Kubernetes: https://kubernetes.io/

[20] docker: https://www.docker.com/