Modeling and Delay Analysis for SDN-based 5G Edge Clouds

Ameen Chilwan and Yuming Jiang
NTNU, Norwegian University of Science and Technology, Trondheim, Norway

***Abstract*—The fifth generation (5G) mobile networks are envisioned to provide connectivity not only to mobile users but also to a wide range of other services such as enhanced mobile broadband (eMBB) and massive Internet of Things (mIoT). In order to meet the diverse requirements of these services in 5G, Software Defined Networking (SDN) has been proposed as an enabling technology for both the core cloud and the edge cloud, in addition to Network Slicing to achieve isolation among services. In this paper, an analytical model is developed for such an SDN-based edge cloud, focusing on the support of two services: eMBB and mIoT. To illustrate the use of the model, delay analysis of a switching node in the edge cloud is presented. The results show the relation between the packet delay and the underlying system parameters, such as slice density, and the impact of the SDN controller on the delay. An implication of the model, analysis and results is that they may be used for network / resource planning and admission control in 5G edge clouds to meet delay requirements of the services.**

***Index Terms*—5G, edge cloud, software defined networking (SDN), network slicing, delay analysis**

I. Introduction

The 5G mobile technology is designed to keep up the pace with the explosion in the number of devices connected, and the services provided, on mobile networks. The number is forecast to reach 12.4 Billion by 2022 and these devices are estimated to generate 77 Exabytes of data per month of which 12% will be carried by 5G networks [1].

The mobile traffic mix has also evolved, from voice-only to mainly voice with little data and now to mainly IP traffic. Similarly, the targeted services have also diversified. Mobile networks are no longer confined to serving human users only. They do and will support, e.g., vehicular networks, sensor networks, e-health and broadcast applications, among many others [2]. Each of these applications has different quality of service (QoS) requirements on the services provided by the mobile network. In 5G, such services include *enhanced mobile broadband* (eMBB) and *massive Internet of Things* (mIoT).

Each of these service categories has different quality of service (QoS) requirements and each of them requires isolation from others even though they are utilising the same underlying resources. The corresponding logical networks are referred to as *Network Slices* in 5G [3]. Besides Network Function Virtualization (NFV), Software Defined Networking (SDN) is a key enabling technology for network slicing, which enables engineering traffic among and across network functions for the corresponding network slices [3]. The 5G architectural proposals advocate SDN to be used in both the core cloud and the edge cloud [2], [4].

The focus of this paper is on investigating the capability of such an SDN-based edge cloud in meeting the QoS requirements of its supported services. For the sake of simplicity and tractability, we focus on two service categories, namely eMBB and mIoT. Given that these two types of services have highly different characteristics and QoS requirements, some scheduling policy between them on the data path is assumed for service isolation / differentiation between them. The aim of the paper is to develop a queueing model for performance analysis of the services in an edge cloud, based on which, delay analysis of a switching node is exemplified. Unlike classical packet-switched networks where data forwarding and traffic routing are integrated at the same node, the data plane and the control plane are separated in SDN, which makes the vast amount of analysis results for the classical computer networks, e.g. [5], no more applicable for SDN networks.

The novel contributions of the paper are as follow. First, a queueing model is proposed for performance analysis of the SDN-based 5G edge cloud that supports, e.g., eMBB and mIoT and adopts some scheduling among them on the data path. The proposed model takes into account the impact of SDN controller on the packet forwarding performance in the cloud: A packet entering an SDN switch finding no flow match will invoke the controller to decide the forwarding rule before being forwarded by the switch. In addition, to demonstrate the use of the model, packet delay analysis of a switching node in the edge cloud is presented. Moreover, numerical results are introduced to show the relation between the packet delay and the underlying system parameters, such as slice density, and the impact of the SDN controller on the delay. An implication of the model, analysis and results is that they may be used for network / resource planning and admission control in 5G edge clouds to meet delay requirements of the services. To the best of our knowledge, this is the first attempt to model SDN in 5G edge clouds with emphasis on having multiple network slices across different service categories, although there has been research on modelling SDN for other uses which will be discussed in Sec. VI.

The rest is organized as follows. In Sec. II, the cloud-based 5G network architecture is reviewed. In Sec. III, a queueing network model is proposed to analyze packet forwarding in an SDN-based edge cloud. In Sec. IV, packet delay analysis of a switching node in the cloud is provided. In Sec. V, numerical results are presented and discussed. In Sec. VI, related work is reviewed. Finally, concluding remarks are made in Sec. VII.

II. The System

In this section, a brief description of the system and its operation is presented. Specifically, the network under study follows the standard 5G cellular architecture which divides the infrastructure into two parts: core network, and radio access network (RAN) (of which transport network is a part) as depicted in Fig 1. Some network functions and services are brought closer to the user and hosted in the edge clouds [4] located in the RAN to meet the 5G service requirements [2], while the main functions are situated in the core cloud.

The radio access network is divided into cells where each cell is served by a fifth Generation Node B (gNB). An edge cloud may cover or support multiple such cells. *To simplify the representation, in this study, we will focus on one cell part of the cloud. This edge cloud "cell"* consists of all the equipment at the base station (gNB) used to provide the services, and the radio resources of the cell. The equipment includes an SDN switch, equipment implementing physical network functions (PNFs), and a micro datacenter hosting virtual network functions (VNFs), to support e.g. network slicing in the edge cloud. Such cell SDN switches are connected to the transport network of the whole edge cloud, which further connects to the backbone network and the core cloud as shown in Fig 1.

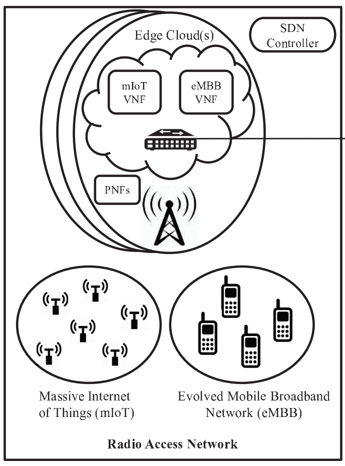

Fig. 1. An overview of the edge cloud in 5G RAN

An SDN switch in this system configuration (i) forwards all the data originating from the mobile terminals and mIoT devices to the respective functions and routes, (ii) provides connectivity to all the edge cloud equipment and functions, and (iii) transports traffic from/to the core network and other edge clouds. The SDN controller for the edge cloud is assumed to be located in the same RAN. *It is also assumed that the control traffic is in-phase with the data traffic*, i.e., the control traffic uses the same channels and occupies the same buffers as the data traffic as provisioned in [6].

To provide communication between the SDN controller and SDN switches, an OpenFlow architecture is considered. Although OpenFlow has a few standard messages, we shall limit our discussion to only those that are used for; i) notifying the controller that a new flow has arrived, ii) installing new rules in the switches, iii) initiating a handover, and iv) configuring the switches. The OpenFlow messages identified for fulfilling these tasks are, *Packet-in*, *Flow-mod*, and *Port-status*.

In SDN, when a packet arrives at a switch, it is checked against the flow table at the switch. If there is a matching rule, the packet is forwarded according to it. Otherwise, a Packet-in *message is generated at the switch and is sent to the controller. Once the controller has determined the forwarding rule, it sends* Flow-mod *messages to all the switches in the route so that they update their flow tables.*

In case of a handover, the OpenFlow-based handover implementation [7] utilizes *Port-status* messages for it. In this approach, the source switch – from where the mobile node is being handed over to the target switch – sends an *Off-port* message to the controller initiating a handover. Subsequently, the target switch sends an *On-port* message to the controller signalling that the mobile node is in its coverage area. Since for each *Off-port* message their is a respective *On-port* message, they can be collectively modelled as *Port-status* messages [7]. The controller, on its part, updates all the switches that are forwarding flows from the mobile node by sending *Flow-mod* messages to them.

It is supposed that multiple classes of services are supported, such as eMBB and mIoT that have highly diverse traffic characteristics. At an SDN switch, some scheduling discipline between the traffic of these service classes is assumed for service isolation / differentiation between them, while within the same class, the traffic shares the same FIFO queue.

III. Queueing Network Model for Edge Clouds

In order to model the system presented in Section II, we refer back to Fig. 1. The notation is summarized in Table I.

A. Traffic to the Switch: Originated and Forwarded

Consider an edge cloud cell with switch i, which spans area $\mathcal{A}_i$, serves a number n_i^e of eMBB slices and a number n_i^m of mIoT slices. It is one of the n cells served by a single SDN controller of the edge cloud. The users in the j^{th} eMBB slice and the devices in the k^{th} mIoT slice are considered to be spatially distributed following Poisson Point Processes with parameters $\alpha_i^{j(e)}$ users per unit area and $\alpha_i^{k(m)}$ devices per unit area respectively as reasoned in [8], [9]. Subsequently, the respective packet arrival processes are given by (1) for eMBB slice and by (2) for mIoT slice where B^e and B^m denote the respective data rates, P^e and P^m stand for their average IP packet sizes, and p^e and p^m are the fractions of active users/devices in the slice at any given time.

$$\zeta_i^{j(e)} = \alpha_i^{j(e)} \ \mathcal{A}_i \ \frac{p^e B^e}{P^e} \tag{1}$$

$$\zeta_i^{k(m)} = \alpha_i^{k(m)} \ \mathcal{A}_i \ \frac{p^m B^m}{P^m} \tag{2}$$

We assume that the spatial distributions for all eMBB and mIoT slices in a cell are i.i.d. across their respective serve classes, also referred to as *verticals* in the rest of the

TABLE I
PARAMETERS AND THEIR DEFINITIONS

Parameter	Description
$\mathcal{A}_i$	Cell size of i^{th} cell
n	Number of cells controlled by an SDN controller
n_i^e	Number of eMBB slices in i^{th} cell
n_i^m	Number of mIoT slices in i^{th} cell
$\alpha_i^{j(e)}$	Users in j^{th} eMBB Slice of i^{th} cell
$\alpha_i^{k(m)}$	Devices in k^{th} mIoT Slice of i^{th} cell
$B^{e/m}$	Data rate of eMBB/mIoT service
$P^{e/m}$	Average eMBB / mIoT packet size
$p^{e/m}$	Percentage of active users in eMBB/mIoT slice
$\zeta_i^{j(e/m)}$	eMBB/mIoT packet arrival rate in j^{th} slice of i^{th} cell
$\lambda_i^{e/m}$	eMBB/mIoT packet arrival rate in cell i
$\gamma_i^{e/m}$	eMBB/mIoT data plane packet arrival rate in cell i
$\Gamma_i^{e/m}$	eMBB/mIoT total (data + control) arrival rate in cell i
Γ_c	Total packet arrival rate in controller
$p_{ji}^{e/m}$	Routing probability of cell i for cell j for eMBB /mIoT
$u_i^{j(e/m)}$	eMBB/mIoT indicator: cell i in new flow path for cell j
$v_i^{j(e)}$	eMBB indicator: cell i in handover path for cell j
$o_i^{j(e/m)}$	Prob. that cell i is in route for cell j for eMBB/mIoT
μ_i	Switch i service intensity
μ_c	Controller service intensity
ρ_i	Switch i server utilization
ρ_c	Controller server utilization
δ_i^e	Handover intensity
$q_i^{nf(e/m)}$	Probability of new eMBB/mIoT flow in cell i
K	Controller buffer size

paper, with parameters ζ_i^e and ζ_i^m respectively. Therefore the aggregate eMBB and mIoT arrival rates in cell i are given by (3) and (4).

$$\lambda_i^e = n_i^e\ \zeta_i^e \tag{3}$$

$$\lambda_i^m = n_i^m\ \zeta_i^m \tag{4}$$

Additionally, the switch of each edge cloud cell also forwards data traffic from other edge clouds. Since the data for each vertical is separated from the other verticals hosted at the switch, the cross data traffic from other edge clouds will go to the queues of their respective verticals. The routing probability from switch j to switch i is given by p_{ji}^e for eMBB and by p_{ji}^m for mIoT vertical. The resulting aggregated traffic input to the eMBB queue at the switch i, is given in (5) and that to the mIoT queue is given in (6).

$$\gamma_i^e = \lambda_i^e + \sum_{j\in N} p_{ji}^e\ \gamma_j^e \tag{5}$$

$$\gamma_i^m = \lambda_i^m + \sum_{j\in N} p_{ji}^m\ \gamma_j^m \tag{6}$$

where N is the set of all switches connected to switch i.

It is worth highlighting that γ_i^e and γ_i^m only consider the traffic in the data plane, and how the traffic from the control plane contributes additionally is presented in the following.

B. Traffic to the Switch: Contributed by the Control Plane

Note that the SDN controller is invoked whenever:

(i) there is a new flow entering the SDN switch;
(ii) a mobile node is handed over from one switch to another.

As a consequence of (i), when a packet enters the switch and *misses* a flow match, it is sent to the controller. For the probability that the arriving packet belongs to a new flow and does not have a match in the switch flow table, we call it *miss* probability, and use $q_i^{nf(e)}$ to denote for the eMBB packets and $q_i^{nf(m)}$ for the mIoT packets. The controller then computes the route for the flow and sends the forwarding rules back to the switch, and all the switches that will forward packets of that flow. This means that *the first packet of a flow goes through the switch twice: first time to the controller and second time being transmitted to the next node*. Whether the switch i is in the route for a new flow originating from switch j and should have a flow added to it is indicated by the function $u_i^{j(e)}$ for eMBB and by the function $u_i^{j(m)}$ for mIoT. If the value of the function is 1 it means that the switch is in the route, while 0 means otherwise.

For (ii), to simplify the representation and taking into account the fact that IoT devices are generally stationary in a broad range of applications, in the following, handovers will be only considered in the eMBB vertical. Suppose that handover requests from eMBB arrive at the gNB, either originated or terminated, with rate δ_i^e requests per second. The requests are received by the gNB and subsequently its switch i sends a port-status message to the controller. In response, the controller sends back a message to the switch to update the flow table. It also sends messages to all the switches whose flow tables should be updated. Accordingly, an indicator function $v_i^{j(e)}$ is used to capture whether a switch needs to be updated in response to a handover. It has the value 1 if switch i needs to be updated for handover originating or terminating at switch j, and 0 otherwise.

Note that the functions $u_i^{j(x)}$, and $v_i^{j(x)}$ where $x \in \{e, m\}$, are similar indicator functions, and the routing probabilities p_{ji}^x from neighbouring switches are similar to the probability that switch i lies in the path for traffic originated from a different switch j. To ease representation, we will use $o_i^{j(x)}$ to collectively model if switch i is in the route for flows coming from an arbitrary switch j, where j may be the same as i. This implicitly means that it will also cover the two indicator functions $u_i^{j(x)}$, and $v_i^{j(x)}$. The simplified total arrival rates to the eMBB and mIoT queues at the switch are now expressed in (7) and (8) respectively.

$$\begin{aligned}\Gamma_i^e = &\left(1 + q_i^{nf(e)}\right)\lambda_i^e + 2\delta_i^e \\ &+ \sum_{j=1, j\neq i}^{n} o_i^{j(e)}\left(\left(1 + q_j^{nf(e)}\right)\lambda_j^e + \delta_j^e\right)\end{aligned} \tag{7}$$

$$\Gamma_i^m = \left(1 + q_i^{nf(m)}\right)\lambda_i^m + \sum_{j=1, j\neq i}^{n} o_i^{j(m)}\left(\left(1 + q_j^{nf(m)}\right)\lambda_j^m\right) \tag{8}$$

C. Traffic to the Controller

The central part of the system is the SDN controller. Following the same discussion in the previous subsection, the traffic to the controller has two contributors (i) and (ii). Essentially, the way in which the controller is incorporated in our model is that when a packet arrives at the switch and does not find a matching flow rule at the switch, it is forwarded to the controller, due to being the first packet of either a newly generated flow (i) or a handover flow (ii).

This brings us to the expression for the input rate at the controller queue, which is given in (9).

$$\Gamma_c = \sum_{i=1}^{n} \left(q_i^{nf(m)} \lambda_i^m + q_i^{nf(e)} \lambda_i^e + \delta_i^e \right) \tag{9}$$

D. Queueing Network Model

In the edge cloud system we consider, each SDN switch has a dedicated queue for each service class and adopts a certain scheduling discipline to schedule packets among the queues. In contrary, the SDN controller only maintains a single FIFO queue to process requests. For the queue at the controller, in order to avoid excessive delay waiting in queue for the requests, a limitation on the queue length may be enforced.

A blueprint of the system thus modelled by a queueing network is depicted in Fig. 2, considering two inter-connected switches i and j and the controller c.

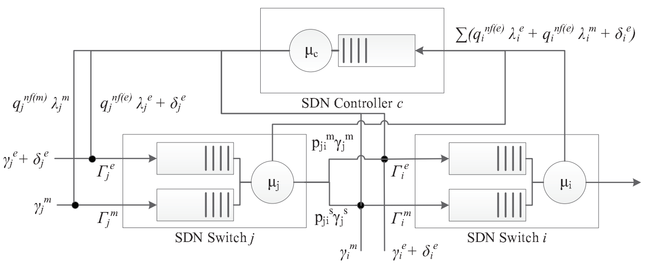
$\sum(q_i^{nf(e)} \lambda_i^e + q_i^{nf(e)} \lambda_i^m + \delta_i^e)$
μ_c
SDN Controller c
$q_j^{nf(m)} \lambda_j^m$
$q_j^{nf(e)} \lambda_j^e + \delta_j^e$
$\gamma_j^e + \delta_j^e$
Γ_j^e
γ_j^m
Γ_j^m
μ_j
SDN Switch j
$p_{ji}^m \gamma_j^m$
$p_{ji}^s \gamma_j^s$
γ_i^m
$\gamma_i^e + \delta_i^e$
Γ_i^e
Γ_i^m
μ_i
SDN Switch i

Fig. 2. Queueing Network Model of the SDN-based Edge Cloud

IV. Delay Analysis

To demonstrate the use of the queueing network model developed in the previous section, this section presents analysis of the packet sojourn time at a switch corresponding to an edge cloud cell. By making use of results in the queueing theory literature, e.g. Jackson network analysis, the single switch delay analysis can be extended to the whole edge cloud case, but due to space limitation, this extension is omitted here.

For the delay analysis, we make the following assumptions. To start with, the system is assumed to support two service classes, namely eMBB and mIoT, and be stable, i.e. $\rho_i = (\Gamma_i^e + \Gamma_i^m)/\mu_i < 1$ and $\rho_c = \Gamma_c/\mu_c < 1$, and in steady state. At the switch, non-preemptive priority scheduling is adopted with eMBB traffic given higher priority, and there is no limitation on the size of each class' queue. As for the controller, a limit on the queue length is enforced, which is denoted as K.

Note that the total traffic to each queue in the switch or the controller is the aggregation of traffic from multiple input processes. For such an aggregate process, following the Palm – Khintchine theorem, we approximate it with a Poisson process. This is in agreement with the literature as argued in [10], [11] for mIoT traffic and [12]–[15] for eMBB traffic.

In addition, for the service time of a packet at the switch or controller, it consists of two parts: processing time for the switch to check if its flow has a matching rule in the switch or for the controller to decide a forwarding rule for its corresponding flow, and transmission time of the packet at the switch or transmission time of the message packet containing the rule to the switch at the controller. To ease the analysis, we assume the service time of a packet is exponentially distributed. This is inline with the Kleinrock independence approximation for classical computer networks [5].

With the above assumptions, the switch is modelled as an $M/M/1/\infty$/priority queueing system, and the controller as an $M/M/1/K$ queueing system. Then, based on existing queueing results for them [16], the respective average number of packets at switch i can be derived as:

$$N_i^e = \Gamma_i^e \frac{(1 + \rho_i - \frac{\Gamma_i^e}{\mu_i})}{\mu_i - \Gamma_i^e} \tag{10}$$

$$N_i^m = \Gamma_i^m \frac{(1 + \rho_i \frac{\Gamma_i^e}{\mu_i} - \frac{\Gamma_i^e}{\mu_i})}{(1 - \rho_i)(\mu_i - \Gamma_i^e)} \tag{11}$$

and the average number of packets in the controller as:

$$N_c = \frac{\rho_c}{1 - \rho_c} - \frac{(K+1)\ \rho_c^{K+1}}{1 - \rho_c^{K+1}} \tag{12}$$

By using (10), (11) and (12), and applying the Little's formula in queueing theory [16], the average time spent in the switch by an eMBB packet is given by $W^e = N_i^e/\Gamma_i^e$ and by an mIoT packet by $W^m = N_i^m/\Gamma_i^m$, and the average time spent by a packet in the controller is given by $W_c = N_c/\Gamma_c$.

Now, we take into account the *miss* probability while calculating the delay due to controller. To ease representation, we assume that the *miss* probability is the same across the vertical, i.e. $q_i^{nf(e)} = q^{nf(e)}\ \forall\ i \in \{1, 2, ..., n\}$ and $q_i^{nf(m)} = q^{nf(m)}\ \forall\ i \in \{1, 2, ..., n\}$.

Finally, the average time taken by an eMBB / mIoT packet in the edge cloud cell is given by (13), where $x \in \{e, m\}$:

$$D_{tot}^x = (1 - q^{nf(x)})W^x + q^{nf(x)}\ (2\ W^x + W_c) \tag{13}$$

In (13), the first term on the right hand side of the equation is for the delay when the packet finds matching flow entry in the switch and hence will only experience one-time W^x at the switch. In addition, the second term is due to that, the packet finds no flow entry in the switch, and consequently it needs to be sent to the controller for it to generate the entry message that is sent to the switch before the packet can use it to be forwarded. As a consequence, the involved delay includes one-time W_c and two times W^x, as also discussed in the previous section.

V. Results and Discussion

We now use the delay analysis (13) and the typical values of the parameters as reported in the literature [2], [12], [14] to obtain results. Specifically, we take $\mu_i = 1250\ pkt/ms$ to show processing at line rate 10 $Gbps$ with average packet length $P_e = P_m = 1\ KB$. In addition, we choose the values $o_i^{j(x)} = 0.2$, $n = 5$, $K = 750$, $\mu_c = 250\ ms^{-1}$, and $\delta_i^e = 0.01\ ms^{-1}$ to emulate the reality as much as possible. Finally, we consider a sub-urban cell covering $\mathcal{A}_i = 1\ km^2$ and having on average $\alpha_i^{j(e)} = 40$ eMBB users, with only $p_e = 30\%$ active at a time, and an average of $\alpha_i^{k(m)} = 400$ active mIoT devices, in each respective service slice. The data rates are kept at 50 $Mbps$ for eMBB and 100 $kbps$ for mIoT following the standards [2].

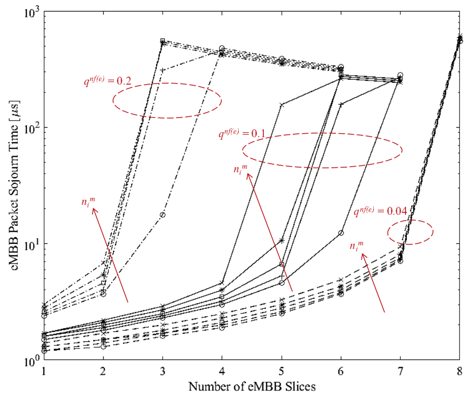

Fig. 3. eMBB Delay Performance for Different Number of Slices

A. Packet Sojourn Time

Fig. 3 studies eMBB packet sojourn time in a single cell with varying number of eMBB slices. The mIoT miss probability is kept constant at $q^{nf(m)} = 0.5$ to show that half the time the packets arriving from mIoT slices will be forwarded to the controller. This value is kept high to emulate the fact that, in may IoT applications, each mIoT device sends packets that are far apart and hence there is high probability that the flow entry will be deleted from the flow table. The packet sojourn time is plotted for three different eMBB miss probability values, i.e. $q^{nf(e)} \in \{0.04, 0.1, 0.2\}$. Here, the first value is determined experimentally in [12], while the other two values demonstrate cases with shorter flow tables for faster lookups. Also, for each $q^{nf(e)}$ value, sojourn time plots are generated for different number of mIoT slices. Thus, for $q^{nf(e)} = 0.04$ results are plotted for $n_i^m = 2, 4, 6, 8, 10$, while in the other two cases the number of mIoT slices are taken to be $n_i^m = 1, 2, 3, 4, 5$.

The results show that by increasing the queries to the controller, there is significant impact on the packet delays. For instance, for the case of 4% miss probability, the packet sojourn time in a cell remains less than 1 ms even when $n_i^m = 10$ and the maximum number of eMBB slices achieved is $n_i^e = 8$ in this case. But, once $q^{nf(e)}$ becomes 0.1, the maximum number of slices reduce to $n_i^e = 7$ for maximum of $n_i^m = 5$. Although sojourn times remain less than 1 ms in this case, but for individual curves abrupt rise in delays are observed. So, when $n_i^m = 1$ the sojourn time jumps an order of magnitude when number of eMBB slices are increased from six to seven. This behaviour is observed due to saturation in controller. The arrival intensity at the controller becomes higher than the service intensity and therefore the delays shoot upwards. Another manifestation of this phenomenon is observed when $n_i^m \geq 3$. The curves seem to flatten at around $0.3ms$ as controller queue will keep overflowing and, consequently, dropping packets.

Finally, for $q^{nf(e)} = 0.2$, the system exhibits a similar behaviour as for 10% miss probability but with controller queue getting saturated as early as at $n_i^e = 3$. This can be seen by flattened curves in Fig. 3.

Another noticeable behaviour of eMBB vertical is that the sojourn times remain fairly close for different number of mIoT slices for a certain miss probability. This is a result of the priority given to eMBB vertical over mIoT in the switch. But as the miss probability increases, the deviation follows the suit because the controller involvement increases and its queue is not prioritized. Furthermore, the rapid saturation of the system by increasing controller-invoking frequency clearly indicates the bottleneck role that the controller plays in this setting.

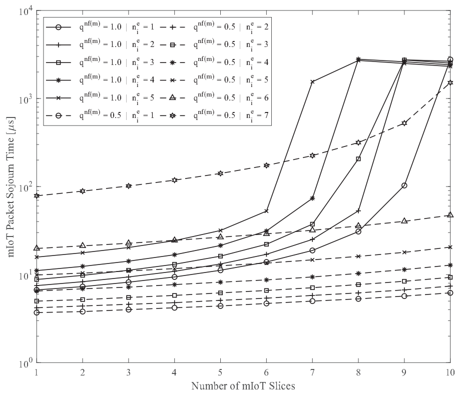

Fig. 4. mIoT Delay Performance for Different Number of Slices

Along the same lines, for mIoT vertical, Fig. 4 shows the packet sojourn time as a function of the number of mIoT slices in a cell for different values of $q^{nf(m)}$ and n_i^e but by keeping constant $q^{nf(e)} = 0.04$. The results are presented for $q^{nf(m)} \in \{0.5, 1.0\}$ and $n_i^e = 1, 2, 3, 4, 5$ for $q^{nf(m)} = 0.5$ and two additional values, $n_i^e = 6, 7$, for $q^{nf(m)} = 1.0$.

The values of $q^{nf(m)}$ are chosen considering that individual mIoT devices send data in very few packets at a certain instant of time and then remain dormant for a long period, typically in minutes. This causes deletion of their flow entries from the switch flow table. The worst case is that each device sends all its data in a single packet, for example environmental sensors, and remains dormant for minutes. It is captured by $q^{nf(m)} = 1.0$ while an optimistic case is given by $q^{nf(m)} = 0.5$.

It is evident from Fig. 4 that the maximum number of mIoT slices under all scenarios presented is $n_i^m = 10$. Unlike the eMBB case, mIoT packets experience greater deviation with increasing n_i^e for the same value of $q^{nf(m)}$. The reason traces back to the priority that eMBB packets have over mIoT packets. It is due to the same reason that mIoT packets experience larger delays even when the system is not loaded. So, for example, in case of a single mIoT slice with seven eMBB slices, the packet delay is already $100\mu s$. Similar to its eMBB counterpart, the mIoT packet sojourn time curve also flattens out when it reaches around $2ms$ indicating the saturation of controller queue.

B. Admissible Region

In order to determine the admissible region, i.e. the maximum combination of the number of eMBB slices and that of mIoT slices which can be supported by a cell, Fig. 5 presents the number of eMBB slices that can be accommodated in the cell provided a certain number of mIoT slices. Here, we use the same parameters values as in Section V-A with $q^{nf(m)} = 1.0$ and $q^{nf(e)} = 0.04$ to represent the worst case for mIoT vertical and a practical case for eMBB vertical.

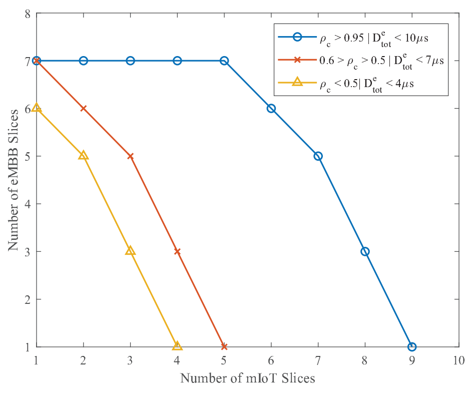

Fig. 5. Number of eMBB Slices under Certain Conditions

The results presented in Fig. 5 also consider the finding from Section V-A that the controller is a delay performance bottleneck. Thus, the results are plotted considering three controller utilization cases, where, the controller is either highly utilized, medium utilized, or mildly utilized. It is seen from Fig. 5, that a larger number of mIoT slices results in a smaller number of eMBB slices if the total delay experienced by a packet is guaranteed. An interesting observation is made in the high utilization case where reducing the number of mIoT slices below 5 does not allow the increase in eMBB slices above 7. The reason for this behaviour is that the switch utilization takes over the controller utilization as the bottleneck.

C. Effect of mIoT Traffic to Controller on eMBB Sojourn Time

In order to further analyse the impact that mIoT vertical has on the performance of eMBB vertical, we further investigate how eMBB vertical behaves with varying conditions of mIoT vertical. The plot in Fig. 6 shows the effect of two mIoT parameters on eMBB sojourn time, namely, the number of

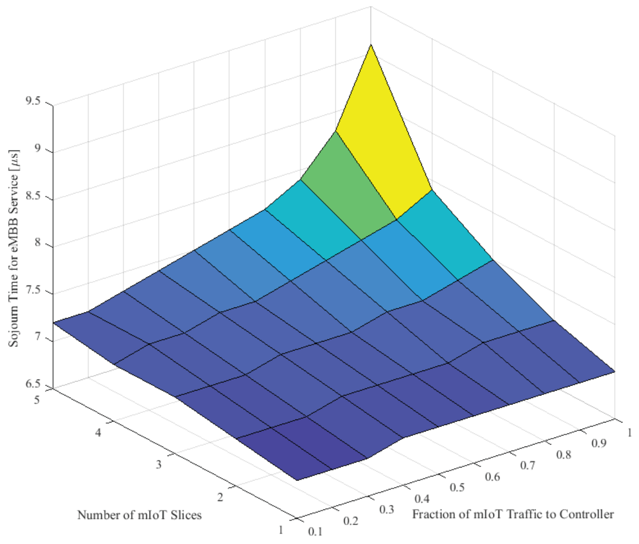

Fig. 6. eMBB Packet Sojourn Time with varying mIoT Vertical States

mIoT slices and the fraction of mIoT traffic forwarded to the controller.

The plot Fig. 6, which is drawn for seven eMBB slices, shows the massive improvement that can be attained in eMBB sojourn time if the expiry time of mIoT flows is just made long enough for at least 10% of the flows to remain intact. The gain in sojourn time is much smaller for lesser number of mIoT slices.

D. Discussion

As shown in Fig. 3 – Fig. 6, under realistic parameter settings in link speed and eMBB and mIoT traffic rates, the average sojourn time at a switch in the edge cloud is in the order of $10\mu s$. Even with propagation delay considered and more accurate traffic and service models in the analysis, when the network diameter of the edge cloud is not too large, e.g. around 5, it can be expected that the average edge-to-edge delay of the edge cloud can be well around $100\mu s$. We believe such a delay range suffices not only the delay requirements of typical mIoT applications but also those of a wide range of eMBB. This is because, the Markov inequality in probability theory tells that the probability that a packet's delay is more than x is not more than $x/E[X]$, where $E[X]$ denotes the mean. For instance, with $E[X] = 100\mu s$, the probability that a packet delay in the edge cloud exceeds $10ms$ is not more than 1%, which is well suitable for a wide range of multimedia applications.

Another observation from Fig. 3 – Fig. 6 is that the maximum numbers of eMBB and mIoT slices that can be supported by an edge cloud cell is rather limited, particularly less than 10 in most cases. We remark that, for simplicity, another 5G service class, which is ultra-reliable low-latency communication (URLLC), has not been factored in the analysis. If URLLC would be taken into consideration, such slice numbers would be further restricted. An implication is that such service / network slices could become a scarce resource and care is needed for planning their use.

VI. Related Work

Since this paper essentially focuses on modelling SDN-based 5G edge cloud, the most relevant research is the one that is conducted in modelling SDN generically, and as a 5G enabler technology specifically. An early related work is presented in [12], where the authors modelled a single SDN node and a single SDN controller as a queueing system with feedback. The controller was modelled as M/M/1-S queue, meaning that it has Poisson arrival and departure processes and limited buffer space, while the switch was modelled with an infinite buffer size. An experiment was used to derive model parameters and the model was used to establish expressions for packet sojourn time and packet loss.

In [13], the authors modelled a single SDN node and a single SDN controller as a Jackson Network. They used the model to quantify packet sojourn time and network throughput. This work was extended for multiple SDN nodes controlled by a single SDN controller in [14]. It was the first time in [14] that *Flow-mod* messages of the OpenFlow standard [6], the de facto standard for SDN controller-switch interactions, were also modelled in addition to the conventional *Packet-in* packets for updating all the switches in a flow route.

As far as modelling mobile edge networks are concerned, the most relevant work is presented in [7] in which the authors used the resilient models from [14] and added the missing mobility feature, namely, handover. The authors, therefore, incorporated *Port-status* messages in addition to the *Packet-in* and *Flow-mod* messages that were previously modelled. Another analytical model for a mobile edge cloud is presented in [15]. The authors of [15] first introduced Follow Me Cloud (FMC) in which services are enabled in the edge clouds closest to the mobile nodes. The modelling technique used followed the assumption that the arrival and service processes are Poisson processes and relevant performance metrics, such as, service migration cost and service disruption time, were derived.

Finally, the idea of using SDNs for sensor networks and IoT, along with sensor OpenFlow and Software Define Wireless Sensor Networks (SDWSNs), was proposed in [17]. Furthermore, authors in [18] proposed a mobile edge cloud architecture that can furnish for IoT networks. Incidentally, in the literature, energy consumption and reservation are used as main performance metrics for SDWSNs and very few attempts exist to quantify packet delays, for example in [19].

VII. Conclusion

In this paper, a queueing network model for performance analysis of SDN-based 5G edge clouds is developed. A novel contribution of this model is to take into account the impact of the data and control plane separation on the traffic to the switch and that to the controller. To demonstrate the application of the model, the average delay of a packet passing through a switch in the edge cloud is analyzed, which supports two 5G service verticals, eMBB and mIoT, and adopts priority to schedule their traffic. By incorporating realistic parameter settings into the delay analysis, numerical results are obtained and discussed. Although for tractability of the analysis, some assumptions on the traffic and service processes are made, the essential implications of the results are mostly revealed, e.g. the order of delay and the admissible region, which, we believe, shed new insights on QoS provisioning in 5G edge clouds.

References

[1] Cisco, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022,” Cisco Public, White Paper, 19 Feb 2019.

[2] NGMN Alliance, “NGMN 5G White Paper,” Next Generation Mobile Networks, White Paper, 17 Feb 2015.

[3] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges,” *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, 2017.

[4] P. Rost, A. Banchs, I. Berberana, M. Breitbach, M. Doll, H. Droste, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi, “Mobile Network Architecture Evolution toward 5G,” *IEEE Communications Magazine*, vol. 54, no. 5, pp. 84–91, 2016.

[5] L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications.* John Wiley & Sons, 1976.

[6] Open Networking Foundation, “OpenFlow Switch Specification Version 1.5.1,” ONF, Tech. Rep., 26 March 2015.

[7] M. Alotaibi, A. Helmy, and A. Nayak, “Modeling Handover Signaling Messages in OpenFlow-Based Mobile Software-Defined Networks,” *Journal of Computer Networks and Communications*, vol. 2018, 2018.

[8] S.-W. Ko, K. Han, and K. Huang, “Wireless Networks for Mobile Edge Computing: Spatial Modeling and Latency Analysis,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5225–5240, 2018.

[9] M. Gharbieh, H. ElSawy, A. Bader, and M.-S. Alouini, “Tractable Stochastic Geometry Model for IoT Access in LTE Networks,” in *IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–7.

[10] K. Doddapaneni, A. C. Tasiran, E. Ever, F. Omondi, P. Shah, L. Mostarda, and O. Gemikonakli, “Does the assumption of exponential arrival distributions in wireless sensor networks hold?” *International Journal of Sensor Networks*, vol. 26, no. 2, pp. 81–100, 2018.

[11] F. Metzger, T. Hoßfeld, A. Bauer, S. Kounev, and P. E. Heegaard, “Modeling of Aaggregated IoT Traffic and its Application to an IoT Cloud,” *Proceedings of the IEEE*, vol. 107, no. 4, pp. 679–694, 2019.

[12] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, “Modeling and Performance Evaluation of an OpenFlow Architecture,” in *Proceedings of the 23rd International Teletraffic Congress*, ser. ITC ’11. International Teletraffic Congress, 2011, pp. 1–7.

[13] A. Chilwan, K. Mahmood, O. N. Østerbø, and M. Jarschel, “On Modeling Controller-Switch Interaction in OpenFlow based SDNs,” *International Journal of Computer Networks & Communications*, vol. 6, no. 6, p. 135, 2014.

[14] K. Mahmood, A. Chilwan, O. N. Østerbø, and M. Jarschel, “Modelling of OpenFlow-based software-defined networks: the multiple node case,” *IET Networks*, vol. 4, pp. 278–284(6), September 2015.

[15] T. Taleb and A. Ksentini, “An Analytical Model for Follow Me Cloud,” in *2013 IEEE Global Communications Conference (GLOBECOM).* IEEE, 2013, pp. 1291–1296.

[16] L. Kleinrock, *Queueing Systems, Volume 1: Theory.* John Wiley & Sons, 1975.

[17] T. Luo, H. Tan, and T. Q. S. Quek, “Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks,” *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, November 2012.

[18] S. Husain, A. Kunz, A. Prasad, K. Samdanis, and J. Song, “Mobile Edge Computing with Network Resource Slicing for Internet-of-Things,” in *IEEE 4th World Forum on Internet of Things (WF-IoT).* IEEE, 2018.

[19] C. B. Margi, R. C. A. Alves, G. A. N. Segura, and D. A. G. Oliveira, “Software-Defined Wireless Sensor Networks Approach: Southbound Protocol and Its Performance Evaluation,” *Open Journal of Internet Of Things (OJIOT)*, vol. 4, no. 1, pp. 99–108, 2018.