Battle of Microservices: Towards Latency-Optimal Heuristic Scheduling for Edge Computing

Amit Samanta* Yong Li‡ Flavio Esposito†
*Department of Computer Science and Engineering, India Institute of Technology, Kharagpur, India.
‡Department of Electronic Engineering, Tsinghua University, China.
†Department of Computer Science, Saint Louis University, USA.

***Abstract*—Edge computing paradigm aims at offloading microservices from mobile devices to the edge of network, reducing latency and increasing computational ability. Differently from the monolithic architecture, in a microservice-based system, an application is developed as a suite of microservices, each running independently on containers. To optimize the offloading decision, existing schemes often require a priori knowledge of the service type, latency-requirement, or both. Such limitations, in turn, hinders the quality-of-service (QoS) for real-time (mobile) applications. This paper presents FLAVOUR, a novel distributed and latency-optimal microservice scheduling mechanism for edge computing platform to achieve minimal service latency, while providing guaranteed transmission rates to minimize microservice completion times (MSCT). To design FLAVOUR, we first formulate a stochastic service delay minimization problem with constraints on MSCT and network stability. By solving this problem, we derive an optimal latency-optimal heuristic scheduling problem, which establishes the theoretical foundation for FLAVOUR. We have implemented a FLAVOUR prototype and evaluated FLAVOUR through both the testbed experiments and CloudSim simulations. Our preliminary results show that FLAVOUR holds great promise in terms of latency and throughput under different traffic dynamics.**

I. Introduction

In recent times, edge computing platforms have gained importance in modern day-to-day mobile applications, as well as for data intensive tasks. Generally, the conventional standalone applications execute their services fully on mobile devices, whereas cloud applications execute their services at the cloud. Mobile cloud applications, instead, have processing and inter-process communication that span cloud processes and mobile devices, jointly [1], [2], [3], [4]. Such applications require high data processing, infrastructures, storage capability that may not be fulfilled on the standalone mobile devices. Conventional clouds are designed to be stationed on a logically centralized datacenter. However, applications being supported by Mobile Edge Computing (MEC) have intermittent communication that may range over long-distance, and the higher service-delay may not fulfill the requirements of real-time mobile applications [5], [6], [7], [8], [9]. The fundamental objective of MEC is to design a small-scale cloud platform deployed at the edge of the network, where different mobile edge devices execute their microservices of different applications. Such platforms are placed in proximity of users to provide seamless and low-latency access to edge services. Now a days, the edge computing platform have adapted a microservice-oriented architecture in order provide different advantages to real-time mobile applications [10], [11]. This kind of infrastructure is designed to minimize the microservice completion times (MSCT). As many of today's edge computing applications, such as traffic monitoring, augmented applications, and retail recommendation, have very demanding latency requirements, and even a small latency can directly affect application performance and degrade user experience. Thus, it is necessary to maintain the quality-of-services among real-time mobile applications.

978-1-5386-9376-6/19/$31.00

Motivation. Designing a novel distributed and microservice scheduling mechanism in MEC is a challenge, as different mobile applications [12], [13] generally have the different resource, computational and latency demands. For example, after a natural or man made disaster, urgent imagery needs to be process at the edge to rescue survival despite the variation of latency of the underlying network infrastructures [13]. Fundamentally, in a mobile edge device, several applications may have different heterogeneous microservices running in the background to fulfill their requirements. They need to be scheduled among edge servers to minimize the MSCT and maximize the throughput. Hence, there could be a situation, where each microservice is trying to complete their tasks first and eventually get involved in a battle among them. Such situations, not only increases the MSCT, but also decreases the throughput and QoS. Therefore, the microservices are required to schedule among edge servers without any a priori knowledge of service type and latency requirements. Motivated by the above discussion, we list our key challenges as follows:

(i) ***Multiple Level Priority Queue***: FLAVOUR contemplates multiple priority queues for microservices at the edge cloud server to implement a multiple level feedback priority queue. Here, a microservice is systematically ousted from higher queues to lower queues based on the number of packets (bytes) it has sent. Consequently, short microservices are probably to be completed in the first few high queues and thus be prioritized over long microservices in general, which enables FLAVOUR to follow shortest job first (SJF) without knowing microservice sizes beforehand.

(ii) ***MSCT Minimization***: The solution must be able to enforce an adaptive and optimal latency-optimal heuristic microservice scheduling. It should minimize average MSCTs for latency-sensitive short microservices, while not adversely affecting the MSCTs of long ones.

Contributions. In this paper, we present FLAVOUR, a novel adaptive latency-optimal heuristic scheduling scheme for MEC, that can provide the correct transmission rates to meet service deadlines and achieve minimal packet latency. When designing FLAVOUR, we explicitly set out our design goals at the network edge without modifying underlying hardware. The key contribution is that FLAVOUR builds up on the theoretical foundation to minimize packet latency and service deadline completion. In particular, we establish a theoretical foundation for latency-optimal heuristic microservice scheduling (§III-A). We formulate a service delay minimization problem with constraints on microservice completion and queue size (§III-B). We then apply the Lyapunov optimization framework to transform this problem to a convex problem (§III-B), so that an optimal solution can be derived for the transformed convex problem. Our analysis confirms the stability and the optimality of the algorithm. We then implemented a FLAVOUR prototype and evaluated the FLAVOUR through both testbed experiments and CloudSim simulations. Our preliminary results show that FLAVOUR holds great promise in terms

of MSCT and throughput. (§IV).

II. Architectural Overview

We present FLAVOUR in Figure 1, a novel latency-optimal heuristic scheduling framework/system based on two key ingredients. First, it uses a set of microservices to estimate their latency requirements using statistical analysis, rather than assuming it as in current approaches [4], [14], [15]. The latency estimated by statistical approach enjoy three important properties: they are low-stretch, diverse, and naturally optimal. Second, it schedules the microservices to edge servers to avoid congestion problem (battle among microservices) in the network. While these ideas have been explored in isolation previously, their combination turns out to be surprisingly powerful, and allows FLAVOUR to achieve near-optimal performance. Our work is the first practical implementation and comprehensive evaluation of this combined approach, called latency-optimal heuristic scheduling. Consequent, we present FLAVOUR, a miroservice heuristic schedul-

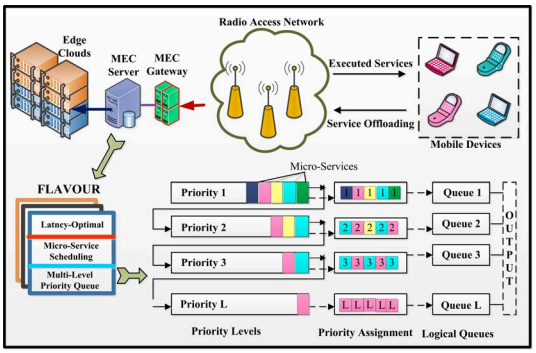


Figure 1: FLAVOUR Architecture

ing mechanism that aims to minimize MSCT by applying Shortest Job First (SJF) on the premise that service size is not known a priori. Therefore, inspired by [16], FLAVOUR leverages multiple priority queues to implement a multiple level feedback queue at the edge server side. In which, a microservice is systematically ousted from higher queues to lower queues based on the number of packets (bytes) it has sent. Consequently, short microservices are probably to be completed in the first few high queues and thus be prioritized over long microservices in general, which enables FLAVOUR to follow shortest job first (SJF) without knowing microservice sizes beforehand. Our evaluation results show that FLAVOUR significantly outperforms existing information-agnostic schemes.

III. Theoretical Modeling

In this section, we discuss the theoretical foundation of the FLAVOUR and design a near-optimal solution.

A. System Model

Consider, a MEC platform with N logical links between edge devices to edge servers, each with a link-capacity of H_n bits per second (bps). Here, the edge devices are the normal mobile devices and edge servers are the servers placed inside the edge cloud. In the network, the total number of active applications is $\mathcal{A}$. At time t, application a transmits a microservice at a rate of $R_a(t)$ bps, and the remaining data size is denoted as $J_a(t)$, and the remaining time till MSCT $G_a(t)$. We consider that each application provides deadline information to the edge server end in the request to send data. The expected rate for application a is defined as, $\zeta_a(t) = J_a(t)/G_a(t)$. We also do not consider the routing of the microservices, and assume that the microservice from application a will be scheduled through a fixed set of logical links $N(a)$. For link n, the aggregated input rate is, $\mathcal{B}_n = \sum_{a\in\mathcal{A}(n)} R_a$, where the set of microservices that pass through link n is, $\mathcal{A}(n)$.

Stochastic Latency Estimation. Service latency should be minimized, as the deadline completion for short and query microservice is very sensitive towards it. Hence, we should obtain low latency for microservices, which may not impose limitation on service flow rate and may not hurt the microservice completion time. Further, resource-hungry edge applications expand for more bandwidth and may build up long queue consequently, increasing the latency. We believe this inherent aggressiveness should be counteracted by considering the long term average of network metrics. Instead of limiting the service flow rate, we decide to use the long term average of per-packet latency as the minimization objective. The latency a microservice experienced on link n with a service load F_n is denoted, $\mathcal{D}_a(\mathcal{B}_n)$. For application a, the average service latency is $(\sum_{n\in N(a)} \mathcal{D}_a(\mathcal{B}_n))/a$. The latency of link n, $\mathcal{D}_a(\mathcal{B}_n)$, is a function of $\mathcal{B}_n$, the aggregated arrival rate at link n. $\mathcal{D}_a(\mathcal{B}_n)$ is a positive, convex and increasing function. We define the objective function as the long term average of the summation of microservice latency and total time a microservice waits in the designed priority queue until it can be executed $\mathcal{Q}_a(t)$.

$$\mathcal{S}_{(R,\mathcal{B})} = \lim_{\mathcal{T}\to\mathcal{T}_{max}} \sum_{t=0}^{\mathcal{T}-1} \sum_{n\in N(a)} \frac{[\mathcal{D}_a(\mathcal{B}_n(t)) + \mathcal{Q}_a(t)]}{\mathcal{T}} \tag{1}$$

Service Rate Stability Here, the long term average service rate is required to be larger than the expected rate. This constraint is an approximation to the real-time edge traffic, where the microservices cannot be infinitely long. It is mathematically defined as: $R_{max}(t) \leq \lim_{t\to t_{max}} \sum_{t=0}^{t'} \frac{1}{t}\big(\zeta_a(t) - R_a(t)\big) \leq 0, \forall a$, where $R_{max}(t)$ denotes the maximum service rate. By incorporating this constraint, we are essentially guaranteeing that, for every microservice that requires ζ_a, the service rate R_a is on average larger than ζ_a, where ζ_a is the expected rate for application a.

Queue Modeling The instantaneous queue length at link n is $Q_a(t)$. The rate stability condition is denoted as, $\lim_{t\to t_{max}} \frac{1}{t} Q_a(t) = 0$. To stabilize the queues, the long-term average of aggregated service rates must satisfy [17], $\lim_{t\to t_{max}} \sum_{t=0}^{t'} \frac{1}{t}\mathcal{B}_n \leq H_n$. Here, the basic consideration is that the long-term average service load generated in the network can be handled by the capacity of the network, so that the network can be stabilized with proper rate control scheme. Otherwise, it will be difficult to avoid service packet loss. This constraint is later relaxed into the objective of the minimization problem during the transformation, so that microservices that use service rates that exceeds link capacity is penalized.

B. Problem Formulation

We aim to formulate an adaptive latency-optimal heuristic scheduling scheme to minimize service delay with guaranteed throughput. A service delay minimization problem is formulated to encapsulate the above latency and network stability constraints. Mathematically,

$$\textbf{Min}\mathcal{S}_{(R,\mathcal{B})} = \lim_{\mathcal{T}\to\mathcal{T}_{max}} \sum_{t=0}^{\mathcal{T}-1} \sum_{n\in N(a)} \frac{[\mathcal{D}_a(\mathcal{B}_n(t)) + \mathcal{Q}_a(t)]}{\mathcal{T}} \tag{2}$$

$$\mathcal{B}_n = \sum_{a\in\mathcal{A}(n)} R_a \tag{3}$$

$$\text{subject to } R_{max} \leq \lim_{t \to t_{max}} \sum_{t=0}^{t'} \frac{1}{t} \big(\zeta_a(t) - R_a(t)\big) \leq 0 \quad (4)$$

$$\lim_{t \to t_{max}} \sum_{t=0}^{t'} \frac{1}{t} \mathcal{B}_n \leq H_n \& R_a(t) > 0 \quad (5)$$

As mentioned above, the link service capacity constraints are relaxed into the objective function to allow for temporary overloading of links. In the rest of this section, we apply the Lyapunov optimization framework to transform this problem to a convex problem, and then derive a near-optimal solution to the transformed convex problem.

C. Transformation Using Lyapunov Optimization

Using the Lyapunov optimization theorem and the drift-plus penalty method [18], the minimization of long term average are substituted with an equivalent convex minimization problem. Here, we rewritten the equation as:

$$W_a(t) = \sum_a \Bigg\{ E \sum_{a \in \mathcal{A}(n)} [\mathcal{D}_a(\mathcal{B}_n(t)) + \mathcal{Q}_a(t)] + Y_a(t)\zeta_a(t)/R_a(t) + Q_a(t)R_a(t) \Bigg\} \quad (6)$$

The optimization problem is mathematically expressed as:

$$\text{Min } W_a(t) \quad (7)$$

$$\text{subject to } \mathcal{B}_n(t) = \sum_{a \in \mathcal{A}(n)} R_a(t) \quad (8)$$

where E is a non-negative weight that is chosen as desired to affect a performance tradeoff. In this transformation, the inequality constraints in 4 need to be transformed into virtual queues, $Y_a(t)$. These queues take the expected rate $\zeta_a(t)$ as input and the actual rate $R_a(t)$ as output. We have, $Y_a(t+1) = [Y_a(t) + \zeta_a(t) - R_a(t)]^+$. Virtual queues $Y_a(t)$ stores the difference in the expected service rate and actual service rate, and the queue lengths are essentially historical deviation from service rates of the microservices. With the transformed problem, we developed an adaptive microservice heuristic scheduling algorithm by greedily minimizing the upper bound of the Lyapunov drift. We have derived the near-optimal solution of the optimization problem.

D. Near-Optimal Solution

By applying the properties of near-optimal solution and the KKT conditions of the optimization problem, we obtain a heuristic algorithm to achieve the optimality for Equation (7).

$$\frac{d}{dt} R_a(t) = F'_a(R_a(t)) - \sum_{a \in \mathcal{A}(n)} \zeta_a(t) \quad (9)$$

where $F_a(R_a(t)) = -Y_a(t)\zeta_a(t)/R_a(t) - Q_a(t)R_a(t)$, $\zeta_a(t) = d'_a(\mathcal{B}_n(t))$ and $\mathcal{B}_n(t) = \sum_{a \in \mathcal{A}(n)} R_a(t)$. We establish the stability and optimality of the service delay optimization problem in Equation (9) by proving the following Lemma 1.

Lemma 1. *Suppose, $\Omega(a)$ is strictly concave in nature.*

$$\Omega(a) = \sum_{n \in N(a)} F_a(R_a(t)) - \sum_{n \in N(a)} \sum_{a \in \mathcal{A}(n)} d'_a\Big(\sum_{n \in N(a)} R_a\Big)$$

The unique solution that a^* maximizes $\Omega(a)$ is a stable point of the dynamic system, to which all microservices converge.

Proof. Let, $F''_a(R_a) = -\sigma''_a(R_a) = -\frac{Y_a(t)J_a(t)/G_a(t)}{R_a^3} < 0$, for $R_a(t) > 0$. As $d_a(\mathcal{B}_n)$ is convex and positive, thus $d''_a(\mathcal{B}_n) > 0$. $\Omega(a)'' = \sum_{n \in N(a)} F''_a(R_a) - \sum_{a \in \mathcal{A}(n)} d_a(\mathcal{B}_n) < 0$, therefore $\Omega(a)$ is a concave function. Also, $d\Omega(a)/dR_a(t) = F'_a(R_a(t)) - d'_a(\mathcal{B}_n(t))$. $F'_a(R_a(t)) = \frac{Y_a(t)J_a(t)/G_a(t)}{R_a^2} - Q_a(t)$ is an decreasing function, and approaches infinity as a goes to 0. On the other hand, $d_a(\mathcal{B}_n)$ is strictly increasing, $d'_a(\mathcal{B}_n) > 0$. It follows that there exist an unique value R_a such that $d\Omega(a)/dR_a(t) = 0$ for $R_a > 0$.

$\Omega(a)$ is therefore strictly concave with an interior maximum. The maximal a^* is unique, and can be identified by $d\Omega(a)/dR_a(t) = 0$.

$$\frac{d}{dt}\Omega(a) = \sum_{a \in \mathcal{A}(n)} \frac{d\Omega}{dR_a(t)} \frac{d}{dt} R_a(t)$$

$$= \Bigg[\sum_{a \in \mathcal{A}(n)} F_a(R_a(t)) - \sum_{a \in \mathcal{A}(n)} \sum_{n \in N(a)} d'_a\Big(\sum_{n \in N(a)} R_a\Big) \Bigg]^2 \geq 0$$

which demonstrates that $\Omega(a)$ is strictly increasing with respect to t, unless $a = a^*$. $\Omega(a)$ is therefore a Lyapunov function of the dynamic system, and the theorem concludes. □

IV. Implementation

To demonstrate the practicality of our approach, we have implemented the proposed heuristic scheduling scheme – FLAVOUR, deployed at the network edge. In this section, we detail each component of our implementation.

FLAVOUR Prototype. To design our proposed scheme, we develop a small-range setup composed of 10 servers, each server comprised of Intel core-i5 processor, 1.7 GHz CPU and 4 GB of memory with DDR3 RAM. All servers run on Ubuntu 14.04 – 64 bit with Linux 3.13.X kernel version. Among 10 servers, 6 were designed for edge services and the other 4 for standard cloud services. We develop a client/server model to generate the several services from different applications and measure the Service Completion Time (SCT) at the application layer. The client application, running on one of the servers, generates different services for the other 10 servers to offload them efficiently. All servers are equipped with a Broadcom 43224AGN Gigabit Ethernet NICs [19]. The servers are connected to each other with a Gigabit Ethernet switch having 144M pps and maximum port bandwidth 96G. This switch supports strict priority queuing with at most 8 classes of service queues. Hence, our system FLAVOUR uses eight priority queues. The queue is selected based on the application types, specifically critical applications (i.e., traffic monitoring or VR) can be directly sent to highest priority queue and non-critical applications (i.e., background updates) will use lowest priority queue. Therefore, in our system, one microservice can be shifted to the higher queue.

Service Assembling. All modern real-time applications are basically designed on top of high-level abstractions using low-level socket APIs with an equivalent real-time traffic monitoring system. Therefore, for layer-wise information passing between high-level and low-level requires refactoring through multiple abstraction and different libraries. Our implementation is based on the CloudSim simulator [20] and leverages a Remote Procedure Call (RPC) protocol. After assembling/collecting different services from several applications, FLAVOUR optimally differentiates among the three service classes. Thereafter, FLAVOUR acquires the information about the services, such as sizes and deadlines, for the computational offloading of services. The information of services are collected from the applications running at the user space. Along with, the information passing of services is still a challenging task for Kernel programming.

A. Evaluation Settings

We evaluate FLAVOUR using different experimental settings and a series of extensive CloudSim [20] simulations.

Experimental Setup. We describe the experimental setup for the performance evaluation of proposed scheme - FLAVOUR. We consider 50 edge devices distributed over an area of 1000 m X 1000 m and one macro base station (MBS) co-located on a MEC server. The MEC server located in the MBS, whose computation capability is 100 GHz and the computation capability of edge device is 0.7 GHz with bandwidth 20 MHz. The capability of the MEC server is 100 GHz and the total number of CPU cycles of computation task is $1,000$ Megacycles. Here, the microservice deadline is varied within $[4000, 6000]$ ms and computation resource demand varied within $[10, 20]$ MHz. The data traffic arrival is modeled as Poisson process $[0, 10]$ unit/sec and the expected size of data traffic is 100 Mbits with service arrival rate within $[0, 10]$ (mean size = 1 Mbit). The cellular backhaul delay coefficient is considered to be 0.0001 sec/KB [14], [21]. The total time duration to offload the computation services of mobile edge devices are randomly distributed between 5 and 10 ms. The corresponding computation file size of each computational service varies within the range 300 and 800 KB. The delay requirements of edge devices is considered to be within the range 0.5 and 1 second.

Traffic Workload. We implement our scheme on 10 servers, each machine configured with Intel core-i5 processor and 1.7 GHz CPU. For this work, we consider two types of traffic workloads – delay-sensitive ($i.e.$, generally edge services) and delay-tolerant ($i.e.$, cloud services) workloads. We give higher priority to the delay-sensitive traffic than the delay-tolerant traffic workloads. Therefore, in our simulation, we ran critical services at a higher priority than the normal and background services.

Benchmarks. To evaluate the performance, we use two benchmarks - MSME [10] and GREEDY. MSME [10] proposed a microservice scheduling scheme for MEC. They designed an architecture that includes heterogeneous machines that can handle different microservices, which tries to assign the optimal resources to microservices and also minimizes the service delay. We also compared with a greedy approach of microservice scheduling, it follows a heuristic solution to find a local optima at each stage with the aim of finding a global optima. For analyzing the proposed scheme, we used a few service utilization metrics to measure the FLAVOUR's accuracy. The throughput is defined as the ratio of the total number of successful bytes received and total number of bytes sent. Finally, the probability of microservice completion rate in this work is defined as the probability of completing a microservice within it's specified microservice completion times. The percentage of the performances are calculated, while taking the average of around 50 rounds. FLAVOUR-Approx is the FLAVOUR scheme without any optimal solution.

B. Results and Discussions

We now examine the microservice completion rate in Figure 2. It can be seen that FLAVOUR outperforms MSME and GREEDY methods for all kinds of microservices. For query and short microservices, all schemes performs similarly under 20 % of maximum load. However, the rates diverge as the traffic load increases. For the query type microservices, MSME-Approx and GREEDY-Method gets the completion rate close to 0.2 with the maximum load of 20 %. For short microservices, the completion rates of the two schemes – MSME-Approx and GREEDY-Method are consistently low and less than 0.3, as compared to around 25 % for FLAVOUR-Opt and 20 % for FLAVOUR-Approx. For long microservices, FLAVOUR again shows advantages over MSME-Approx and GREEDY-Method. Even with the loads increase to maximum, more than 40 % of long microservices meet their deadlines for FLAVOUR. Further, MSME-Approx and GREEDY-Method cannot provide fair resources to long microservices with various deadlines properly using latency-optimal heuristic scheduling, resulting in 30 % and 40 % deadline missed at 80 % of maximum load. As shown across Figure 2, FLAVOUR is able to meet almost all microservice deadlines, and to achieve network stability under different dynamics of loads, as a result of FLAVOUR's ability to find and sustain the right rate for every microservice. However, MSME-Approx and GREEDY-Method are not able to precisely capture the rate requirements of microservices.

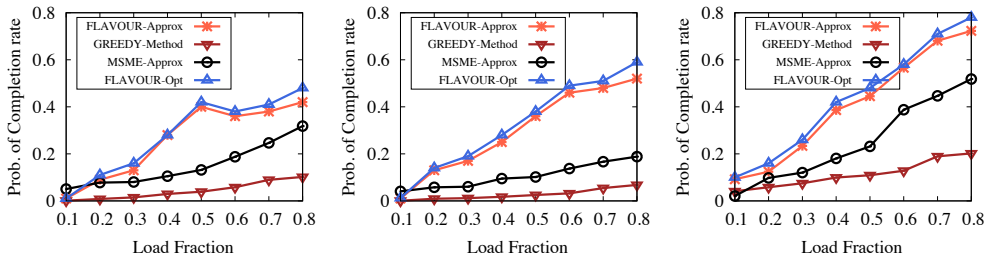
In Figure 3, we compare the throughput of FLAVOUR for three types of microservices. Here, the throughput is defined as the amount of data packets transmitted successfully within the microservices deadlines. The line for FLAVOUR-Opt is the total amount of data packets sent from the devices for different traffic load levels, which is also denoted as the "*Traffic Threshold*". The performance of FLAVOUR-Approx is close to FLAVOUR-Opt, i.e., the traffic threshold for all types of microservices. Further, MSME-Approx and GREEDY-Method can achieve at most 28 and 20 % of maximum throughput compared to FLAVOUR-Approx for the maximum traffic load of more than 20 %. These two methods fall off at high load levels for all types of microservices. As for the overall performance across all types of microservices, FLAVOUR-Approx satisfies almost all traffic requirements for having lower completion time.

V. Related Work

Prior works have studied resource optimization and offloading mechanism for MEC platform, they are discussed as follows. Fillip *et al.* [10] proposed a microservice scheduling scheme for MEC. They designed describing an architecture that includes heterogeneous machines that can handle different microservices, which tries to assign the optimal resources to microservices and also minimizes the service delay. Wang *et al.* [2] proposed an online resource allocation scheme for mobile edge devices without requiring any a priori knowledge on either the resource price or the user mobility. You *et al.* [22] proposed an energy-efficient resource allocation scheme for computational service offloading in MEC. On the other hand, Fernando *et al.* [23] proposed a work sharing algorithm for MEC to offload the computational services at the nearby mobile devices. Chen *et al.* [3] proposed an efficient offloading scheme for multi-user computational offloading in MEC. Mueller *et al.* [24] proposed a platform for continuous computing offloading from cloud to edge. Mao *et al.* [25] proposed a dynamic computational offloading with energy harvesting mobile edge devices in MEC. Liu *et al.* [26] proposed a multi-user computation offloading algorithm using game theoretic approach in MEC. Differently than these existing approaches, we propose a latency-optimal heuristic scheduling scheme for microservices in MEC. As the existing works are problematic in handling microservices with specified deadlines and also providing the optimal latency based on the demand rate of the microservice under different network dynamics. Further, they can not accommodate the different types of traffic and their dynamics.

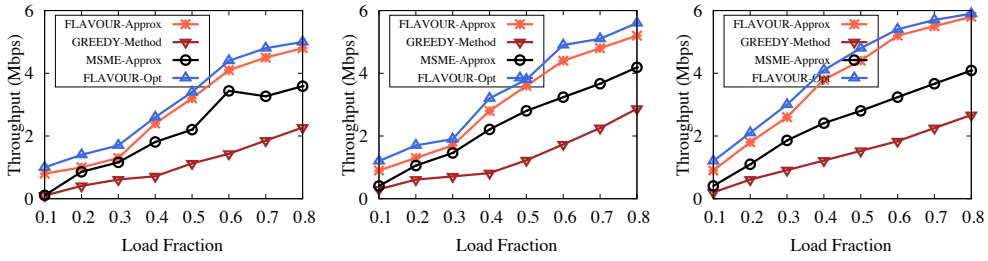
VI. Conclusion

We presented FLAVOUR, a novel distributed and microservice scheduling mechanism for edge computing platform to achieve minimal service latency, while providing guaranteed transmission rates to minimize microservice completion times (MSCT). To design FLAVOUR, we first formulated a stochastic service delay minimization problem with constraints on service completion time and network stability. By solving this problem, we derived an optimal latency-optimal heuristic scheduling problem, which establishes the



(a) Short Microservices (b) Query Microservices (c) Long Microservices

Figure 2: Numerical simulation results: completion rate



(a) Short Microservices (b) Query Microservices (c) Long Microservices

Figure 3: Numerical simulation results: throughput

theoretical foundation for FLAVOUR. We have then implemented a FLAVOUR prototype and evaluated FLAVOUR through both testbed experiments and CloudSim simulations. Our preliminary results show that FLAVOUR holds great promise in terms of latency and throughput. We envision the design of a latency-oblivious scheduling scheme for microservice-oriented edge clouds and also design a microservice load balancing scheme under different network dynamics.

VII. Acknowledgment

This work has been partially supported by NSF awards CNS-1647084 and CNS-1836906.

References

[1] A. Samanta and Y. Li, "Time-to-think: Optimal economic considerations in mobile edge computing," in *IEEE INFOCOM Workshops*, 2018.

[2] S. Wang *et al.*, "Online Placement of Multi-Component Applications in Edge Computing Environments," *IEEE Access*, 2017.

[3] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM TON*, vol. 24, no. 5, pp. 2795–2808, 2016.

[4] A. Samanta and Y. Li, "DeServE: Delay-agnostic Service Offloading in Mobile Edge Clouds," in *ACM/IEEE SEC*, 2017, pp. 24:1–24:2.

[5] P. Garcia Lopez *et al.*, "Edge-Centric Computing: Vision and challenges," *ACM SIGCOMM CCR*, vol. 45, no. 5, pp. 37–42, 2015.

[6] M. Satyanarayanan *et al.*, "Edge Analytics in the Internet of Things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.

[7] Crutcher et al., "Hyperprofile-Based Computation Offloading for Mobile Edge Networks," in *IEEE MASS*, 2017, pp. 525–529.

[8] F. Esposito, A. Cvetkovski, T. Dargahi, and J. Pan, "Complete Edge Function Onloading for Effective Backend-Driven Cyber Foraging," in *Proceedings of IEEE WiMob*, 2017, pp. 1–8.

[9] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Computing Surveys. (to appear)*, vol. abs/1806.06191, Oct 2018.

[10] I. D. Filip, F. Pop, C. Serbanescu, and C. Choi, "Microservices scheduling model over heterogeneous cloud-edge environments as support for iot applications," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[11] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Computing*, vol. 3, no. 6, pp. 76–83, 2016.

[12] W. Hu *et al.*, "Quantifying the Impact of Edge Computing on Mobile Applications," in *ACM APSys*, 2016, pp. 5:1–5:8.

[13] Ventrella et al., "Load profiling and migration for effective cyber foraging in disaster scenarios with FORMICA," in *NetSoft 2018, Montreal, QC, Canada, June 25-29*, 2018, pp. 80–87.

[14] K. Zhang *et al.*, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access*, 2016.

[15] A. Samanta and Z. Chang, "Adaptive Service Offloading for Revenue Maximization in Mobile Edge Computing with Delay-Constraint," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[16] W. Bai *et al.*, "Information-Agnostic Flow Scheduling for Commodity Data Centers," in *USENIX NSDI*, 2015, pp. 455–468.

[17] L. Chen *et al.*, "Towards minimal-delay deadline-driven data center tcp," in *ACM HotNets*, 2013.

[18] M. J. Neely *et al.*, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, 2005.

[19] M. Chowdhury and I. Stoica, "Efficient Coflow Scheduling Without Prior Knowledge," in *ACM SIGCOMM*, 2015, pp. 393–406.

[20] R. N. Calheiros *et al.*, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Soft.: Practice and exp.*, 2011.

[21] A. Samanta and Y. Li, "Latency-Oblivious Incentive Service Offloading in Mobile Edge Computing," in *IEEE/ACM SEC*, 2018, pp. 351–353.

[22] C. You *et al.*, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Tran. on Wireless Comm.*, 2016.

[23] N. Fernando *et al.*, "Computing with Nearby Mobile Devices: a Work Sharing Algorithm for Mobile Edge-Clouds," *IEEE Tran. on Cloud Com.*, vol. PP, no. 99, pp. 1–1, 2016.

[24] H. Mueller *et al.*, "Poster Abstract: Continuous Computing from Cloud to Edge," in *IEEE/ACM SEC*, 2016, pp. 97–98.

[25] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE J. on Selected Areas in Comm.*, vol. PP, no. 99, pp. 1–1, 2016.

[26] Y. Liu *et al.*, "A Multi-user Computation Offloading Algorithm Based on Game Theory in Mobile Cloud Computing," in *IEEE/ACM SEC*, 2016.