Poster: Latency-Oblivious Incentive Service Offloading in Mobile Edge Computing

Amit Samanta*† Yong Li†
*Department of Computer Science and Engineering, India Institute of Technology, Kharagpur, India.
†Department of Electronics Engineering, Tsinghua University, Beijing – 100084, China.
E-mails: {amit.samanta049@gmail.com, liyong07@tsinghua.edu.cn}

***Abstract*—We design a latency-oblivious incentive service offloading scheme to manage complex network services for future mobile services. We build our prototype and show its feasibility in terms of latency and total incurred cost by using mobile edge computing as an example use case in a realistic testbed.**

I. Introduction

Over the years, Mobile Edge Computing (MEC) [1], [2] platform have acquired a lot of attention among researchers, as it provides effective computational offloading ability at the network edge. Inherently, it supports a prolific and incentive computational service offloading scheme, while tuning several application- and network-level optimization. Fundamentally, MEC units are deployed at the network edge to optimize the end-to-end latency of computational services. They are presumed to benefit the offloading procedure, while decreasing the service latency between the mobile users and edge servers. The core element of any mobile computation offloading framework is the designing an optimal decision engine, since it determines when a service needs to be offloaded to an external (MEC) server. Offloading a service to an external server may incur expensive overhead, hence the offloading decision shall be based on predictions of the latency and total time required to offload, as well as the computation time among other possible metrics. It is not easy to predict the service latency of an application method ahead of its execution. An offloading framework addresses these challenges to make efficient offloading decisions and also to provide application developers and/or users a way to integrate their application into the offloading framework. Most offloading frameworks proposed so far predict the available bandwidth or execution time [3]. They have completely ignored the variation in latency requirements of mobile edge services, which is termed as *latency-oblivious*. Another common assumption is that the inputs of the computational services do not vary much, which can lead to imprecise estimation of latency requirements. However, modern mobile application requires different processing powers and latency capabilities to process their services at local operator clouds, private edge clouds (also called MEC hosts) and remote clouds. Therefore, it is necessary to accurately estimate the latency-requirement of mobile users. Hence, it is important to propose a latency-oblivious incentive service offloading scheme for MEC platform.

This poster presents an incentive service offloading framework, while taking into consideration of optimal latency requirements of mobile users. We present a novel latency-oblivious incentive service offloading scheme (LO-ISO). It aims at maximizing users profit, while optimally estimating latency-requirements of different user-specific applications.

II. Latency-Oblivious Service Offloading

As previously discussed that the mobile devices have a very stringent latency requirements to offload the computational services. However, it is very tough to figure out the actual latency requirements of mobile devices practically. Also, the latency requirements of mobile devices change dynamically, as it is oblivious to the edge platform. Hence, first, we need to estimate the total service latency encountered by mobile devices, while offloading the computational services. Later, we propose an incentive service offloading scheme, while taking into consideration of unique priorities of different services.

***Latency Approximation**:* To estimate the total service latency, we present a latency approximation scheme.

• *Execution Latency*: The execution latency $\mathcal{D}_i^{exe}(t)$ is depended on the average waiting time and total network delay of incoming and outgoing user requests [4]. Mathematically, $\mathcal{D}_i^{exe}(t) = \mathbb{T}_i(S_i) + \mathbb{T}_n et(i)$. Here, $\mathbb{T}_i(S_i)$ and $\mathbb{T}_n et(i)$ denote the average waiting time and total network delay of incoming and outgoing service S_i.

• *Queuing Latency*: The service queuing delay $\mathcal{D}_i^{que}(t)$ in the network for available services at time t is: $\mathcal{D}_i^{que}(t) = \mathcal{G}\mathbb{Q}_i(t)$ [4]. Here, $\mathbb{Q}_i(t)$ denotes the average queuing latency and $\mathcal{G}$ denotes the number of existing services in the queue.

• *Offloading Latency*: The offloading latency $\mathcal{D}_i^{off}(t)$ is directly proportional to the total waiting time to offload the computational services of users. It is defined as: $\mathcal{D}_i^{off}(t) = \frac{\mathbb{J}_i^t}{q_i^t}$ [4]. Here, $\mathbb{J}_i^t$ and q_i^t denote the total service length and time to execute the service S_i at time t, respectively. Hence, the total estimated service latency $\mathcal{D}_i^{tot}(t)$ experienced by a user is the addition of both service execution latency $\mathbb{D}_{EL}^t$ and service offloading latency $\mathbb{D}_{off}^t$, which is expressed as: $\mathcal{D}_i^{tot}(t) = \mathbb{T}_i(S_i) + \mathbb{T}_n et(i) + \mathcal{G}\mathbb{Q}_i(t) + \frac{\mathbb{J}_i^t}{q_i^t}$. The total estimated service latency $\mathcal{D}_i^{tot}(t)$ may not be the actual latency of users, as it is dynamic and changes at each time instant. Thus, we design a latency changing factor, which captures the rate of change of latency at each time instant. Mathematically:

$$\Theta_i^t = f\left(\frac{d^n\mathcal{D}_i^{tot}(t)}{dt^n}, \frac{d^{n-1}\mathcal{D}_i^{tot}(t)}{dt^n}, \cdots, \frac{d\mathcal{D}_i^{tot}(t)}{dt^n}, \mathbb{D}_i^t\right) \quad (1)$$

Based on this, we estimate the actual latency requirement at time t (where $\mathbb{D}^t_{act} = \Theta^t_i \mathcal{D}^{tot}_i(t)/\delta t$), we have,

$$\mathbb{D}^t_{act} = \max\left\{\Theta^t_i \mathcal{D}^{tot}_i(t)/\delta t, \mathbb{D}^{t-1}_{act}\right\} \quad (2)$$

Incentive Service Offloading Framework: After estimating the total service latency, now we present an optimization model for service offloading in MEC. The optimization problem aims at maximizing the economical benefit for users and services, as services are exploring their resources to execute different request from users. Thus, in return, they want some kind of incentive from the edge platform. Hence, we design an incentive service offloading scheme, while taking into consideration of total service latency. We formulate a joint optimization problem for incentive service offloading in MEC.

$$\text{Max} \overbrace{\sum_{t\in\mathcal{T}}\sum_{i=1}^{N}\mathcal{U}^t_i}^{\text{utility value}} = \sum_{t\in\mathcal{T}}\Big[\underbrace{\sum_{i=1}^{N}\sum_{j=1}^{M}(\mathcal{P}^t_{off}I_{ij})}_{\text{data offloading cost}} + \underbrace{\sum_{i=1}^{N}\mathcal{P}^t_O I_{ij})}_{\text{placement cost}} - \underbrace{\sum_{i=1}^{N}\frac{\mathbb{D}^t_{act}}{\mathbb{D}^t_{th}}}_{\text{latency factor}}\Big], \quad (3)$$

$$\text{subject to} \quad I_{ij} \geq I^{th}, i \in N, j \in M, \quad (4)$$

$$\sum_{t\in\mathcal{T}} \mathcal{C}^t_{off} \geq \mathcal{C}^{th}_{off}, \mathcal{C}^t_O \geq \mathcal{P}^{th}_O, \quad (5)$$

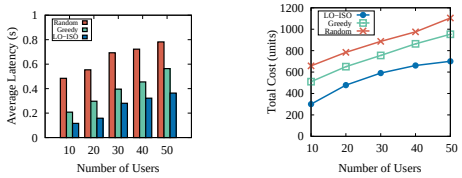
$$\mathbb{D}^t_{act} \leq \mathbb{D}^t_{th}, i \in N, t \in \mathcal{T}, \quad (6)$$

where N and M denotes the number of services and users, $\mathcal{C}^t_O$ denotes the service placement cost, $\mathbb{D}^t_{act}$ and $\mathbb{D}^t_{th}$ denote the actual and threshold service latency, respectively. Equ. (3) represents the joint optimization problem of the economic model. The service flow I_{ij} is to be greater than the service traffic flow as shown in Equ. (4). Equ. (5) represents the offloading cost $\mathcal{C}^t_{off}$ is be greater than the threshold cost $\mathcal{C}^{th}_{off}$. Equ. (6) represents the actual latency requirement $\mathbb{D}^t_{act}$ is to be lesser than the threshold latency requirement $\mathbb{D}^t_{th}$. $\mathbb{D}^t_{th}$ denotes latency requirement. Specific to the designed integer program, we try to model an efficient heuristics for LO-ISO. This only needs resolving one integer liner program (ILP) and which provides less complexity than existing offloading model[1]..

III. Initial Prototype/Results

We have built an initial prototype and emulated the entire LO-ISO system and MEC infrastructure by software packaging. This system may not show the overall performance advantage of the proposed integrated MEC platform, it does show its functionalities and performance for local operations. For evauation, we develop a small-range setup, which composed of 10 servers, each server comprised of Intel core-i5 processor, 1.7 GHz CPU and 4 GB memory with DDR3 RAM. The servers run on Ubuntu 14.04 – 64 bit with Linux 3.13.X kernel version. However, among 10 servers, 6 of them are designed for edge services and other 4 are designed for normal cloud services. We develop a client/server model to generate the several services from different applications and measure the service completion time (SCT) on the application layer. The client application, running on 1 server, generates different services for the other 10 servers in order to offload them efficiently. The servers are equipped with a Broadcom 43224AGN

[1]We compare LO-ISO against other schemes - random (randomly generated service latencies using uniform distribution) and online greedy approach (here it follows a heuristic solution to find a local optima at each stage with the aim of finding a global optima.)



(a) Latency (b) Cost

Fig. 1. Experimental results of LO-ISO

Gigabit Ethernet NICs [5]. 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 class of service queues. We implement LO-ISO in 10 servers, each machine configured with Intel core-i5 processor and 1.7 GHz CPU. We consider both delay-sensitive and delay-tolerant traffic workload. To generate the workloads, we consider the Poisson process with mean 5 and 10 for both delay-sensitive and delay-tolerant user requests, respectively.

Experimental Setup: We consider 50 mobile users and one macro base station (MBS) co-located to a MEC server. The MEC server located in the MBS, whose computation capability is 100 GHz and the computation capability of mobile device is 0.7 GHz. The cellular backhaul delay coefficient is considered to be 0.0001 sec/KB [6], [7]. The total time duration to offload the computation services of mobile edge devices are randomly distributed between 5 and 10 ms. We vary $\mathcal{C}^t_{off}$ and $\mathcal{C}^t_O$ within $[10, 25]$ and $[5, 15]$, respectively. The corresponding computation file size of each computational service varies within the range 300 and 800 KB. The delay requirements of mobile devices is considered to be within the range 0.5-1 s. We envision to test it in a large-scale environment.

Performance Evaluation: Fig. 2(a) shows the average latency in offloading computational services from different applications of mobile users. The lower latency of mobile user guarantees efficient offloading of computational services using LO-ISO, which leads to higher fairness of the system. We also analyze the total cost, as LO-ISO helps the mobile users to offload their services efficiently, while optimizing two main cost factors. Hence, LO-ISO achieves lower cost as shown in Fig. 2(b). We also compared our scheme with the existing Greedy and random approaches. From the figures, we observe that the LO-ISO outperforms the existing schemes.

IV. Future Work

As future work, we intend to fully implement and explore designs for further improving performance and answering the above mentioned questions. This will be the actual complete deployment of the LO-ISO with distributed MEC infrastructure on a large cluster in an edge cloud platform. This would allow us to evaluate its performance with massive data traffic. We would also like to extend a realtime stream framework to support dynamic re-configurations of running MEC stream applications (e.g., auto-scaling scalability, etc.).

REFERENCES

[1] L. Tong, Y. Li, and W. Gao, "A Hierarchical Edge Cloud Architecture for Mobile Computing," in *INFOCOM*, 2016, pp. 1–9.

[2] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-Centric Computing: Vision and challenges," *ACM SIGCOMM CCR*, pp. 37–42, 2015.

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

[4] A. Samanta, Z. Chang, and Z. Han, "Latency-oblivious distributed task scheduling for mobile edge computing," in *IEEE GLOBECOM*, 2018, pp. 1–7.

[5] M. Chowdhury and I. Stoica, "Efficient Coflow Scheduling Without Prior Knowledge," in *SIGCOMM*, 2015.

[6] K. Zhang *et al.*, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access*, pp. 5896–5907, 2016.

[7] A. Samanta and Y. Li, "Time-to-Think: Optimal Economic Considerations in Mobile Edge Computing: Poster," in *INFOCOM*, 2018.