Priority-Based Offloading and Caching in Mobile Edge Cloud

S. Islam, F. Narin Nur, N. Nessa Moon, A. Karim, S. Azam and B. Shanmugan

***Abstract*—Mobile Edge Computing (MEC) is relatively a novel concept in the parlance of Computational Offloading. MEC signifies the offloading of intensive computational tasks to the cloud which is generally positioned at the edge of a mobile network. Being in an embryonic stage of development, not much research has yet been done in this field despite its potential promises. However, with time the advantages are gaining growing attention and MEC is gradually taking over some of the resource-intensive functionalities of a traditional centralized cloud-based system. Another new idea called Task Caching is emerging rapidly with the offloading policy. This joint optimization idea of task offloading and caching is relatively a very new concept. It has been in use for reducing energy consumption and delay time for mobile edge computing. Due to the encouraging offshoots from some of the current research on the joint optimization problem, this research initiative aims to take the progress forward. The work improves upon the "prioritization of the tasks" by adopting a very practical approach discussed forward, and proposes a different way for task offloading and caching to the edge of the cloud, thereby bringing a significant enhancement to the QoS of MEC. We propose a novel priority-based offloading and caching model considering the joint optimization of offloading and caching along with the local computing policy. The simulation results show that our proposed model is able to reduce more delay time and energy cost of MEC than any other model proposed earlier.**

***Index Terms*—caching, computation offloading, mobile edge computing, priority, energy efficient, delay efficient.**

I. INTRODUCTION

NOWADAYS the application of mobile phones are ubiquities in our life. People use this technology for verities of purpose ranging from professional activities to personal, as well as for entertainment. The applications that are being used by the consumers are constantly getting computationally intensive and resource hungry, thereby pushing the smart mobile devices to a cliffhanger as these devices are unable to provide resource and computational support beyond a certain limit. The user experience thus becomes sluggish and inefficient. By offloading [1]-[4], these applications can hand over some of the complex tasks to the remote cloud through a wireless network, which can bring down the energy needs in a substantial scale, but there is always an issue with the delay time, that is, the transmission time from SMD (smart mobile device) to the remote cloud may increase the delay time. Mobile edge computing [1], [2] can have a positive impact on such issues. It can lessen the delay time and simultaneously cater for high performance by offloading the task at the edge of the cellular network.

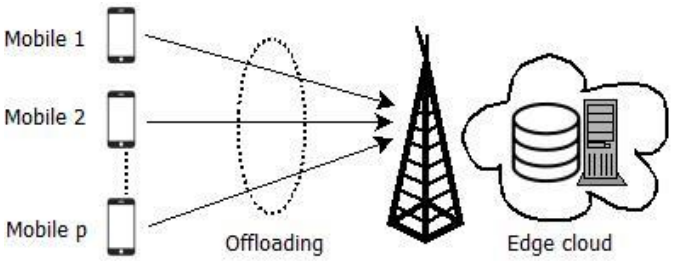


Fig. 1. Illustration of task offloading to the edge cloud.

Mobile edge computing can also overcome the bottleneck of limited computational power as it has available resources. It can also discourse the problem of delay time, often instigated by remote cloud server. On the other hand, Task Caching [1], [5], [6] is relatively a new idea which is used to cache the offloading task so that the repeated offloading of the same tasks can be avoided. Nevertheless, the fundamental concern of MEC remains the joint optimization of offloading and caching [1] and to construct a structure which is efficient to the consumer, meaning it will diminish the energy cost and delay time concurrently. So, we tried to find a model to solve the mentioned issues and make MEC more effective and efficient.

Researchers have worked on multiple energy and delay efficient techniques for MEC systems [1], [14], [15]. In some of the previous works, the authors have proposed and simulated designs to solve the joint optimization problem of caching and offloading to make MEC system more user efficient. The authors of [15] proposed a solution called suboptimal solution (SOS) for MEC. It's a cache assisted offloading scheme to reduce the energy cost of MEC system. They tried to minimize

Manuscript received January 25, 2019; revised April 4, 2019. Date of publication April 25, 2019. Date of current version June 3, 2019. The associate editor Prof. Claudia Canali has been coordinating the review of this manuscript and approved it for publication.

S. Islam, F. Narin Nur and N. Nessa Moon are with the Dept. of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh (e-mails: saiful15-5379@diu.edu.bd, {narin, moon}@daffodilvarsity.edu.bd).

A. Karim, S. Azam and B. Shanmugamare with the College of Engineering, IT and Environment, Charles Darwin University, NT, Australia (e-mails: {asif.karim, sami.azam, bharanidharan.shanmugam}@cdu.edu.au).

Digital Object Identifier (DOI): 10.24138/jcomss.v15i2.707

the computation complexity in a way that would also able to reduce the energy cost and delay time of smart mobile device (SMD). In [14], the authors aim to provide a new caching strategy called femto-caching along with offloading strategy to reduce more energy cost and delay time of MEC system. Femto-caching caches the computing task data which causes the lowest of total energy cost, until caching capacity of edge cloud is over. The authors of [1] proposed a model on joint optimization of offloading and caching which is called TCO to decrease the delay time and energy cost of SMD all together. Their research exhibited fairly a decent outcome and the graphical illustration of their result shows that they were able to reduce an obvious amount of energy cost and delay time.

However, there are few more shortcomings in the previous works, for instance, more clarity needs to be infused on the issues of selecting tasks to cache first, tasks to compute locally and tasks that are to be offloaded at the edge cloud. They did not consider any specific formula to prioritize the computing tasks. It is also a challenging issue to make an efficient offloading decision with task caching (caching and offloading strategy). All these objectives also need to be decided in the light of reduced energy consumption and decreased delay time [1]. Researchers have also worked on multiple techniques for the secure and efficient techniques data transmission over cloud-based systems [9]-[12]. Though previous experiments showed prodigious results, our goal is to solve the above mentioned problems in a near-optimal fashion.

In this paper, our research proposes a novel model that includes task priority. Priority will be calculated from the four main concerns related to the tasks as the number of requests for the task (task popularity), deadline, data size and required resources for the computation. In the (III-B) section, priority factor calculation will be a significant ace in solving caching and offloading problem. By measuring Priority, it can be decided which task to cache first and which task to execute locally, as well as which task to offload at the edge cloud. Implementing the above plan itself has problematic questions to answer, such as priority-based task caching, priority-based task offloading to the edge cloud and to solve the joint optimization problem [1]; and keeping in mind the efficiency factor. As Task Offloading and Task Caching Scheme are being studied to reduce the total energy consumption and the delay time, the goal will be to augment the efficiency markedly.
This paper has been organized in an intuitive way. The related works on MEC has been discussed in the next section and the proposed system model in the following section, that includes system scenario, priority calculation, computation model and problem formulation. Then the proposed scheme is discussed in Section IV followed by performance setup and measurement in Section V. Finally, the paper is concluded in Section VI with some future work directions.

II. Related Works

As the concept of mobile edge computing is growing rapidly, a number of works have been carried out to implement and improve the concept of MEC. Some of the recent works on MEC are discussed here in this section. With the theory of MCC [3], [4], [12] there comes the idea of a network architecture with a centralized cloud [7], [8]. But for the centralized cloud, there are some issues like long latency and mobile traffic growth which affect the concept of mobile cloud computing. These problems can be solved by using mobile edge computing (MEC) [1], [2] which has a network architecture of a distributed cloud [7], [8]. Further recent works on MEC for both multi-user [5], [7] and single user [9] have been identified. For multiuser, distributed [7], [8] computation policy is used to save energy and for low latency using game theory; but multiuser MEC system at a single cloud is more complicated than single user policy. For a single user, energy and latency minimization are derived from optimized offloading and local computing.

Computation offloading is one of the major studies in both MCC and MEC. Computation offloading policy [1-4] is designed to decrease both energy consumption of SMDs and delay time. Algorithms have been consulted to find a more premier way to offload tasks from an SMD to edge cloud. Besides, there are some issues surfaced such as the limitation of resources to support all the computing tasks. It is impractical to think that edge cloud has enough computing resources for all the computing tasks. A major research work on computational offloading has been studied considering multi-server (or distributed server) [5] and multi-user [5], [7] policy. Computational caching [1], [6], [8] is implemented in MEC to cache the computing tasks, data or results to the edge cloud in a view to saving energy and computation time. By the help of computation caching, users do not need to offload the same tasks repeatedly. Another caching strategy is content caching [7], [8]. With the aid of content caching, users can cache the popular content to the edge cloud to reduce the energy cost and delay time. A lot of studies have been done on caching to improve the caching strategies. But the main concern about caching is, 'the capacity of edge cloud storage' still lingers. It is obvious that the storage capacity of edge cloud is limited.

For Computation offloading, both computing resource capacity and storage capacity of edge cloud have to be considered. But for caching policy, only the storage capacity of edge cloud is considered. In that concern, there comes the concept of joint optimization [1], [14], [15] of offloading and caching to the edge cloud. Optimizing the offloading and caching is the best way to reduce the energy consumption and delay time of SMDs. In some of the previous works, the authors have proposed and simulated designs to solve the joint optimization problem. In [15] the authors have proposed a suboptimal solution (SOS) for optimizing the offloading and caching policies. Their proposed solution optimizes the communication, computation and caching policies for a cache-assisted multi-user MEC system to reduce the energy cost and computation delay of user's SMD. They tried to propose a low-complexity solution to reduce the computational complexity. In [14] the authors proposed a strategy called task femto-caching and offloading (TFO) for MEC system. Femto-caching uses storage capacity instead of backhaul capacity at the small cell access points. The authors proposed two femto-caching techniques called uncoded femto-caching scheme which is a special covering problem and coded femto-caching which is a

convex program. They used file popularity distribution technique to optimize the cache allocation at the edge cloud and to reduce the total delay time of all users. The authors of [1] aim to jointly optimize the offloading and caching of tasks to reduce energy consumption and delay time of SMDs. In order to do so, they proposed a scheme called task caching and offloading (TCO) for the MEC system. It is the most recent work on offloading and caching for MEC as per our knowledge. TCO scheme is basically a combination of two sub-problems: one is a convex problem to make the offloading decision and the other one is 0-1 programming to make the caching decision. They consider the joint optimization problem of offloading and caching as mixed-integer nonlinear optimization problem. The simulation result shows that their proposed scheme decreases a significant amount of energy consumption.

III. System Model

The cardinal objective is prioritizing the tasks that needed to be cached or offloaded. For this purpose, five tasks have been computed (T1, T2, T3, T4, and T5). Let us consider each task has different priorities based on their task popularity, deadline, data size, and required computing resource. Our aim is to cut the energy consumption of the SMD and lower the delay time keeping in mind the user's satisfaction. In this paper, three problems, given below, will be addressed:

- Determining the task to cache first: The measured priority will determine the order of the task, that is, the task with the highest priority will be cached first.
- Determining the task to offload first: The highest priority, after due measure, will be offloaded first. In fig. 1, the task offloading scenario of the mobile phone to the edge cloud is demonstrated.
- Determining the task to execute locally or to offload: With this segment, an efficient technique will be presented about the locally executing task or offloading to the edge cloud. The task with the most priority will be executed locally. The level of energy consumption is not being considered here. Because the reduction of delay time effectively means giving the user maximum satisfaction by providing the first task as early as possible.

System model will be discussed briefly in different sections given below.

A. System Scenario

Let's consider there are P number of SMDs and Q amount of tasks to be executed, which can be mentioned as P = (1, 2…P) and Q = (1, 2…Q). Also $M_{p,q}$ has been defined as user **p** requests task **q**. Considering the heterogeneity, three parameters model for computing task has been assumed.
So, $M_{p,q}$= $\{Cr_q, S_q, Dl_p\}$. Here, Cr_q= required computing resource for the task $M_{p,q}$ (in CPU cycles per bit). S_q= data size (in bits). Dl_p= Deadline (in sec).

Besides, another parameter the task popularity has been defined Tp_q(number of request for the task). For calculating purpose, only the first three parameters will be used. But for priority calculation, all four parameters will be considered.

Now, for the system model, only one edge cloud is considered. Here are two things needed to be declared. Cache size and computing capacity of edge cloud. Cache size of edge cloud is defined as C^s and computing capacity (available resource) of edge cloud is defined as C^c.

B. Priority Scheming

To calculate the priority Pr_q, four parameters that effects task computation will be used. To give the user maximum satisfaction possible and considering the energy consumption and delay time four parameters (task popularity, deadline, data size and required resource) will be used to prioritize the computing task.

But for easier calculation purpose and for better priority calculation, weights will be attached to the parameters. Two equal weights w_1 and w_2 are considered. The concept has been demonstrated briefly here.

$$Pr_q = w_1 Tp_q + w_2 \frac{1}{Cr_q + Dl_p + S_q} \quad (1)$$

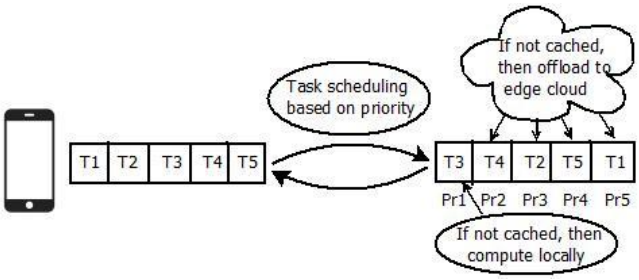


Fig. 2. Illustration of task scheduling based on the task priority calculation of SMD.

Here, in equation (1), task priority Pr_q is calculated using four main parameters related to the task. To provide the maximum user satisfaction, the highest weight is given to the task popularity Tp_q. The other parameters (Cr_q, S_q, Dl_p) are also measured in such a way that it can help us to prioritize the task to reduce delay time and energy cost of user's SMD. In fig. 2, task scheduling is shown based on the priority scheme as per equation (1) to reduce delay time and energy cost of user's SMD. There is also shown that if the task is not already cached at edge cloud than the task with the most priority will be executed locally, otherwise the task will be offloaded to the edge cloud for execution.

The task with the maximum priority will be ranked the highest and to make sure that user gets the task first which user requested for the most. As it also can be seen that bigger the data size the less the priority, that is because it causes more energy consumption and takes more time to offload or execute. The main reason for putting a priority on the task is to give the user the task first which he or she requires the most.

C. Computation Model

Computation model consists of uplink data rate, which is the rate to send data to edge cloud from SMDs. The data rate is denoted as R_p. And it has been formulated as,

$$R_p = B \log_2\left(1 + \frac{P_p H_p}{\sigma^2}\right) \quad (2)$$

Here, B is channel bandwidth, P_p is transmission power for user p, H_p is channel gain for user p and σ^2 is noise power. Using equation (2) uplink data rate is being calculated. Here, downlink data rate won't be considered because after computing at the edge cloud the size of the data become smaller than the one before computing [5]. The downlink data rate is much faster than the uplink data rate. So, it has been ignored.

In computation model section, the calculation of delay time and energy cost for local computing in III-D, offloading in III-E and caching in III-F are discussed briefly.

TABLE I
VARIABLES ASSIGNED FOR PRIORITY-BASED TASK OFFLOADING AND CACHING MODEL

Variables	Assigned to
P	number of SMD
Q	amount of task to be executed
$M_{p,q}$	defined as user p requests task q
Cr_q	required computing resource for the task $M_{p,q}$
S_q	data size
Dl_p	Deadline
Tp_q	task popularity (number of request for the task)
C^s	Cache size of edge cloud
C^c	computing capacity (available resource) of edge cloud
Pr_q	Task priority
w_1	Weight 1
w_2	Weight 2
R_p	data rate
B	channel bandwidth
P_p	transmission power for user p
H_p	channel gain for user p
σ^2	noise power
ε	energy consumption per computing cycle
k	energy coefficient
$\epsilon_{p,q}^{local}$	energy cost for local computation of task $M_{p,q}$
$\epsilon_{p,q}^{cloud}$	energy cost for edge cloud computing of task $M_{p,q}$
$\epsilon_{p,q}$	Total energy cost
μ_p^{local}	CPU computing capability of p
μ_p^{cloud}	computation resource of edge cloud required for p
$\tau_{p,q}^{local}$	execution time for executing task $M_{p,q}$ locally
$\tau_{p,q}^{trans}$	transmission time from SMD to edge cloud of task $M_{p,q}$
$\tau_{p,q}^{cloud}$	execution time of task $M_{p,q}$ at edge cloud
$\tau_{p,q}^{queue}$	queuing time of task $M_{p,q}$
$\tau_{p,q}^{process}$	processing time of task $M_{p,q}$ at edge cloud
$\tau_{p,q}$	total delay time
a	integer decision variable for caching
b	integer decision variable for offloading

D. Energy and Delay Calculation for Local Computing

Our main target is to reduce the energy consumption for computing application task locally and also to reduce delay time for the task execution. Though it is proposed a way to do that but it must be calculated the energy consumption and execution time for executing task locally or by the edge cloud. It is known that a lot of energy used to execute the task but some amount of energy also needed to transmit the data from SMD to edge cloud. Same goes for the execution time. On the basis of earlier research [4], the energy consumption per computing cycle is $\varepsilon = k\mu^2$. Here, k is defined as energy coefficient. It depends on the system architecture. According to the previous research in [11], the value of k is set as 10^{-25}. So, to calculate the energy consumption, the energy cost is obtained for local computation $\epsilon_{p,q}^{local}$ of task $M_{p,q}$ as,

$$\epsilon_{p,q}^{local} = k(\mu_p^{local})^2 Cr_q \tag{3}$$

Here, μ_p^{local} is CPU computing capability of p SMD. Again the execution time $\tau_{p,q}^{local}$ for executing task $M_{p,q}$ locally can be calculated as follows,

$$\tau_{p,q}^{local} = \frac{Cr_q}{\mu_p^{local}} \tag{4}$$

E. Energy and Delay Calculation for Offloading

As equation for energy and delay calculation for the task executing locally, equation for mobile edge computing also needs to be formulated. Thus, the idea is to calculate energy cost of SMD, so the energy cost for edge cloud will not be considered. Energy which is needed to offload or transmit the task data to the edge cloud will only be calculated. And for delay time, the processing time needed by edge cloud to process a task data, the queuing time and transmission time to offload task data from SMDs to edge cloud are need to be calculated. The energy cost for edge cloud computing $\epsilon_{p,q}^{cloud}$ of task $M_{p,q}$ can be denoted as,

$$\epsilon_{p,q}^{cloud} = P_p(\tau_{p,q}^{queue} + \tau_{p,q}^{trans}) = P_p(\tau_{p,q}^{queue} + \frac{S_q}{R_p}) \tag{5}$$

Here, $\tau_{p,q}^{trans}$ is transmission time from mobile device to edge cloud for task q of user mobile device p. $\tau_{p,q}^{queue}$ is the queuing time. At the time of transmission p, tasks need to wait in the queue. This waiting time has been termed as "queuing time".

$$\begin{aligned}\tau_{p,q}^{cloud} &= \tau_{p,q}^{queue} + \tau_{p,q}^{trans} + \tau_{p,q}^{process} \\ &= \tau_{p,q}^{queue} + \frac{S_q}{R_p} + \frac{Cr_q}{\mu_p^{cloud}}\end{aligned} \tag{6}$$

Here, $\tau_{p,q}^{cloud}$ is the execution time for executing task $M_{p,q}$ at the edge cloud, $\tau_{p,q}^{process}$ is processing time, needed by edge cloud and μ_p^{cloud} is the computation resource of edge cloud required for the user mobile device p.

F. Energy and Delay Calculation for Task Caching

Task caching is a new idea that has contributed a lot to the reduction of energy consumption and delay time of SMDs. Task data are cached at the edge cloud for future use which can reduce a lot of energy cost and delay time. First, SMD sends a request to the edge cloud about the task data which needs to be

processed. If the task data are already cached at the edge cloud then the edge cloud informs the SMD that the task data are already in the edge cloud. So, the SMD doesn't need to send the same task data again and again. So, the delay time calculation for the task will only be the processing time $\tau_{p,q}^{process}$ needed by the edge cloud. So there will not be any energy calculation by the SMD as SMD does not need to transmit the task data to the edge cloud. Then the edge cloud sends the result to the SMD, thereby reducing a lot of delay time and energy cost.

But for the task caching strategy, there are some issues like which task to cache first. As it was mentioned earlier that a new strategy is proposed called priority-based task caching which can be used to effectively optimize this issue. But in the case when there is no task is cached, the energy cost and delay time of SMD p will be the same as $\epsilon_{p,q}^{cloud}$ and $\tau_{p,q}^{cloud}$.

G. Problem Formulation

To formulate the equation for execution time and energy cost for mobile edge computing, it is needed to define some decision variables. To measure the task caching problem, $a_q \in \{0,1\}$ is considered as integer decision variable for caching problem. If $a_q = 0$ then it means that task q is not cached at edge cloud or if $a_q = 1$ then it means task q is already cached at edge cloud. So, $a_q \in \{0,1\}$ can be represented as task a_q is already cached at edge cloud and task $1 - a_q$ is not cached at edge cloud, so it can be represented as $a = a_1, a_2, \ldots . a_q$. Same goes for the offloading problem. For offloading, $b_p \in \{0,1\}$ is considered as integer decision variable for the offloading issue. If $b_p = 0$ then it means task q is going to offload to the edge cloud or if $b_p = 1$ then it means task q will be executed locally. So, $b_p \in \{0,1\}$ can be represented as task b_p will be executed locally and $1 - b_p$ task will be offloaded to edge cloud, so it can be represented as $b = b_1, b_2, \ldots . b_p$.

Following the above discussion and considering (caching, offloading and local computing) for $M_{p,q}$ (task q of SMD p) the total delay time can be expressed as,

$$\tau_{p,q} = a_q \frac{Cr_q}{\mu_p^{cloud}} + (1 - a_q)[(1 - b_p)\tau_{p,q}^{cloud} + b_p \tau_{p,q}^{local}] \quad (7)$$

Total energy cost of SMD can be denoted as,

$$\epsilon_{p,q} = (1 - a_q)[(1 - b_p)\epsilon_{p,q}^{cloud} + b_p \epsilon_{p,q}^{local}] \quad (8)$$

Here, it can be seen that there is no energy calculation showed in equation (8) for processing of cached data. Because as it is discussed earlier in section III-F, there is no consumption of energy for SMD during the processing of cached data. As our goal is to reduce energy cost and delay time of SMDs, so there are some conditions need to be true for task q of SMD p. So it can be expressed as,

$$\text{minimize a,b} \quad \sum_{p=1}^{P} \frac{\tau_{p,q}}{Pr_{p,q}} \quad (9)$$

$$\text{s.t.} \quad \sum_{q=1}^{q} a_q s_q \le C^s$$

$$\sum_{p=1}^{P} b_p \mu_p^{cloud} \le C^c$$

$$\tau_{p,q} \le Dl_p$$

$$a_q \in \{0,1\}$$

$$b_p \in \{0,1\}$$

This five constraint conditions and the objective function will give us the minimal energy consumed by the SMD n. The first condition is the size of data which needs to be cached cannot be bigger than the edge cloud capacity. The second condition describes that the required resource cannot overcome the resource capacity of edge cloud. The third condition ensures the task duration should not exceed the deadline. The forth condition shows the decision variables are binary variables (0, 1). And the last condition shows that the task can be separated.

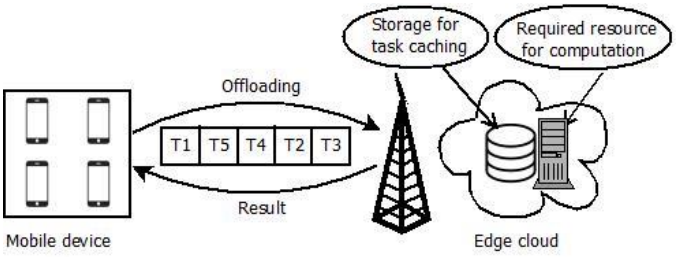


Fig. 3. PrO scheme (priority-based task offloading and caching for MEC to reduce delay time and energy cost of SMD).

Thus our main target is to assure user satisfaction by minimizing the delay time and energy cost of SMD, a priority-based task offloading and caching scheme is proposed to meet the target. In fig. 3, our priority-based task offloading and caching scheme (PrO) is shown, where the tasks Q of user SMD P are prioritized in an efficient way to reduce the delay time and energy cost of SMDs. In equation (9), $\sum_{p=1}^{P} \frac{\tau_{p,q}}{Pr_{p,q}}$ is the formula that shows that higher the output of this formula higher the priority will be for a task. The task with the highest value will be executed locally. The other task will be offloaded to the edge cloud in the decreasing order of the output of equation (9).

IV. Delay Efficient Task Offloading and Caching

The equation (9) $\sum_{p=1}^{P} \frac{\tau_{p,q}}{Pr_{p,q}}$ where, it is shown that the minimization of delay time depending on the decision variables of task offloading and caching a, b. Now both joint optimization problem will be considered as two sub problems. Where first, the optimization of energy efficient task offloading in IV-A will be shown and in the second section, the optimization of energy efficient task caching in IV-B will be discussed.

A. Delay Efficient Task Offloading

For delay efficient task offloading scheme, the decision variable of task caching $a = a^0$ that means it is considered, the delay efficient task caching as optimal solution a^0. Now the objective function becomes the convex optimization problem of b. It can be shown as,

$$F(b) = \sum_{p=1}^{P} a_q^0 \frac{Cr_q}{\mu_p^{cloud}} + (1 - a_q^0)[(1 - b_p)\tau_{p,q}^{cloud} + b_p \tau_{p,q}^{local}] \quad (10)$$

Now, the conditions for the optimization problem of delay efficient task offloading can be obtained as,

$$\begin{aligned} \text{minimize } b \quad & \frac{F(b)}{Pr_{p,q}} \\ \text{s.t.} \quad & \sum_{p=1}^{P} b_p \mu_p^{cloud} \le C^c \\ & \tau_{p,q} \le Dl_p \\ & b_p \in \{0,1\} \end{aligned} \tag{11}$$

Now from here, the optimal solution of delay efficient task offloading b^* can be measured by using optimization method like interior point method.

B. Delay Efficient Task Caching

Now for delay efficient task caching, the solution of delay efficient task offloading will be the optimal one $b = b^*$. So, the objective function becomes the function of a. The optimal solution can be obtained by using branch and bound algorithm which can convert the problem into (0-1) integer programming problem. So the objective function can be shown as,

$$G(a) = \sum_{p=1}^{P} a_q \frac{Cr_q}{\mu_p^{cloud}} + \left(1 - a_q\right)\left[\left(1 - b_p^*\right)\tau_{p,q}^{cloud} + b_p^* \tau_{p,q}^{local}\right] \tag{12}$$

Now, the conditions for the optimization problem of delay efficient task caching can be obtained as,

$$\begin{aligned} \text{minimize a} \quad & \frac{G(a)}{Pr_{p,q}} \\ \text{s.t.} \quad & \sum_{p=1}^{P} b_p \mu_p^{cloud} \le C^c \\ & \tau_{p,q} \le Dl_p \\ & a_q \in \{0,1\} \end{aligned} \tag{13}$$

From here, the optimal solution of delay efficient task caching a^0 can be measured.

Now, the estimated optimal solution of joint optimization problem of delay efficient task offloading and caching can be measured by understanding the linear iterative algorithm.

V. Performance Setup and Measurement

As mentioned earlier in the section III-A, for number of users P, Q number of computation intensive tasks need to be executed. For that, one edge cloud is considered having available resources for task computation at the edge of the cellular network. SMD uses the wireless channel to offload the task to the edge cloud. So, it is assumed that the wireless channel gain as H_p which is defined as H_p= 127+30×logd. Here d is the distance between the user SMD and the edge cloud. It is also needed to assume the transmission power P_p and bandwidth B of SMD to offload the task to the edge cloud are 0.5 W and 20 MHz. Besides, the equivalent noise power is assumed as σ^2=2 × 10^{-13}. Moreover, it is assumed that the computing capacity of SMD and edge cloud are 1 GHz and 25 GHz. Task popularity Tp_q is calculated from the number of request for the task using Zipf distribution. By using probability distribution: normal distribution and uniform distribution [12], the data size S_q and required computing resource Cr_q for the task $M_{p,q}$ are assumed. Cloudsim simulator [13] is used to compare the outcome of our proposed priority-based task offloading and caching scheme (PrO) and the other schemes like task femto- caching and offloading (TFO) scheme [14], suboptimal solution scheme (SOS) [15] and task caching and offloading (TCO) scheme [1].

A. Operative Offloading and Caching Assessment

The strategy of task caching, offloading and local computing for priority-based task offloading and caching (PrO) and the consequences of their action are briefly discussed below. It will help to understand the working process of the proposed priority-based task offloading and caching scheme (PrO).

Consequences of task caching: To measure the priority-based task offloading and caching scheme (PrO), it is needed to measure the task caching policy. The effect of task caching depends on the condition applied in the caching strategy. The task which offloaded first will be cached first at the edge cloud. The offloaded tasks will be cached as per priority of tasks until reaching the cache capacity of edge cloud. Before offload or local execution of every task it will be checked whether the task is already cached or not at edge cloud.

Consequences of task offloading: The task data will be offloaded to the edge cloud after checking whether the task data is already cached or not. Task data will be offloaded to the edge cloud based on the priority of the task. The tasks with higher priorities will be offloaded first (except the task with the highest priority which will be executed locally). For task offloading to the edge cloud, the tasks should fulfil the conditions mentioned in section III-G or task will be computed locally instead of offloading to the edge cloud.

Consequences of local computing: For local computing, the task with the highest priority will be computed locally after checking whether the task is already cached or not. If the task with the highest priority is already cached than the next task with the maximum priority will be computed locally. The reason of exhausting MEC is to compute the intensive task at the edge cloud instead of SMD to reduce delay time and energy cost of SMD. But in this paper, it is considered that the task with the highest priority will be computed locally instead of offloading to the edge cloud to get more user satisfaction. It may cause some energy loss of SMD but it is delay efficient. Besides, the task which doesn't fulfil the conditions mentioned in section III-G will be computed locally instead of offloading to the edge cloud.

B. Performance Evaluation

In performance evaluation, the delay time and energy consumption of user's SMD P for different task data size, edge cloud cache size, required computation capacity per task and number of users has been presented in graphical figure (fig. 4; fig. 5). It will be compared between task femto-caching and offloading (TFO) scheme [14], suboptimal solution scheme (SOS) [15], task caching and offloading (TCO) scheme [1] and priority-based task offloading and caching scheme (PrO). In fig. 4: the energy consumption of user SMD for different task data size, edge cloud cache size, required computation capacity per task and number of user will be shown. And in fig. 5, the delay

time of user SMD for different task data size, edge cloud cache size, required computation capacity per task and number of user will be shown.

Suboptimal solution scheme (SOS): Suboptimal solution is a low complexity cache assisted offloading scheme to reduce the energy cost of MEC system. For that, three parameters for each task have been considered: size of the task data, required computing resource (workload) and size of the computation result. Besides, it also considers the task popularity and randomness in requirements of the task.

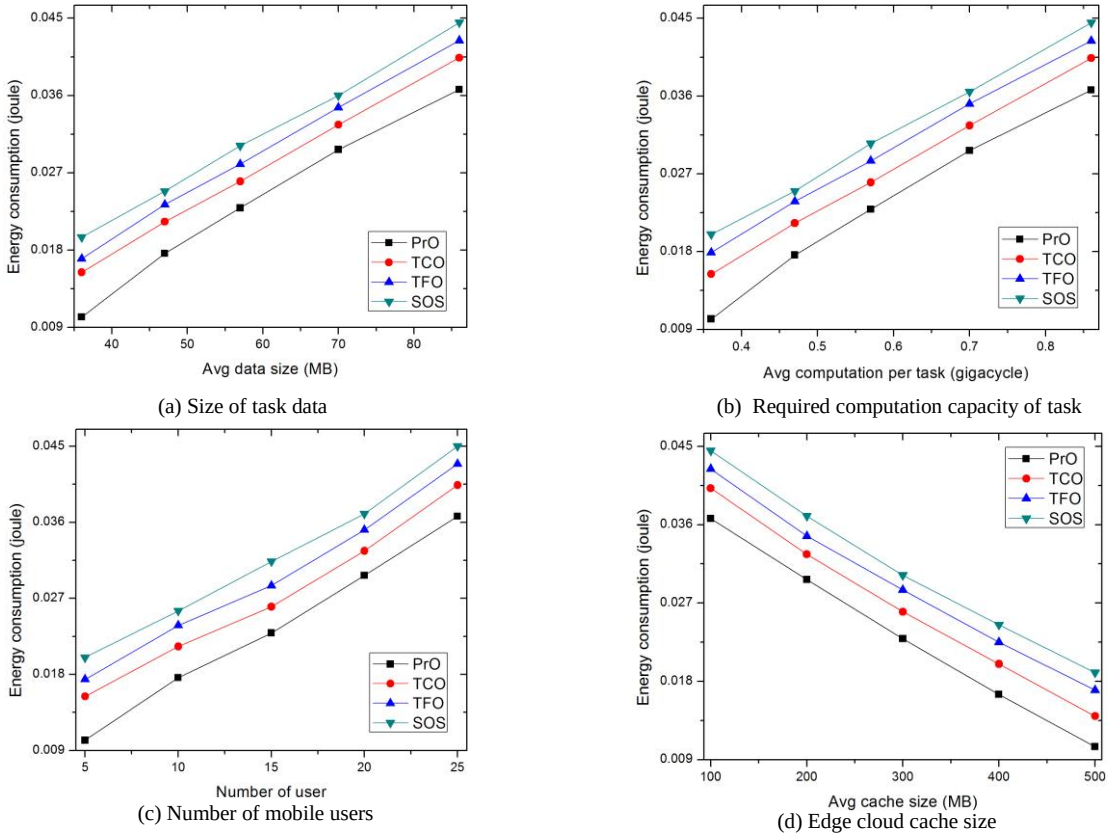
Task femto-caching and offloading (TFO): For TFO scheme, it considers the task data size, required computing resource and task popularity to a certain level. Caching capacity of edge cloud is set as void at the beginning. Femto-caching caches the computing task data which causes the lowest of total energy cost. Until caching capacity of edge cloud is over, cache the task data iteratively.

Task caching and offloading *(TCO)*: In TCO scheme, tasks are cached, offloaded and computed locally considering task popularity, data size and required resource but it doesn't maintain any certain strategy or formula to order the tasks. Though, the simulation result of TCO scheme showed a great result in terms of minimizing the energy cost and computation delay of user SMD.

Priority-based task offloading and caching (PrO): In this paper, priority-based task offloading and caching is proposed to assure more user satisfaction. In PrO scheme, the tasks are cached, offloaded and computed locally maintaining the order of the tasks based on their priority. This (PrO) scheme is able to reduce more delay time and energy cost than any other schemes proposed earlier.

C. Impact of PrO Scheme on Energy Consumption and Delay Time

The impact of priority-based task offloading and caching (PrO) on energy consumption and delay time is compared with task femto-caching and offloading (TFO) scheme, suboptimal solution scheme (SOS) scheme, and task caching and offloading (TCO) scheme. In fig. 4 and fig. 5, it is clear that our PrO scheme has better impact than the other schemes.



(a) Size of task data

(b) Required computation capacity of task

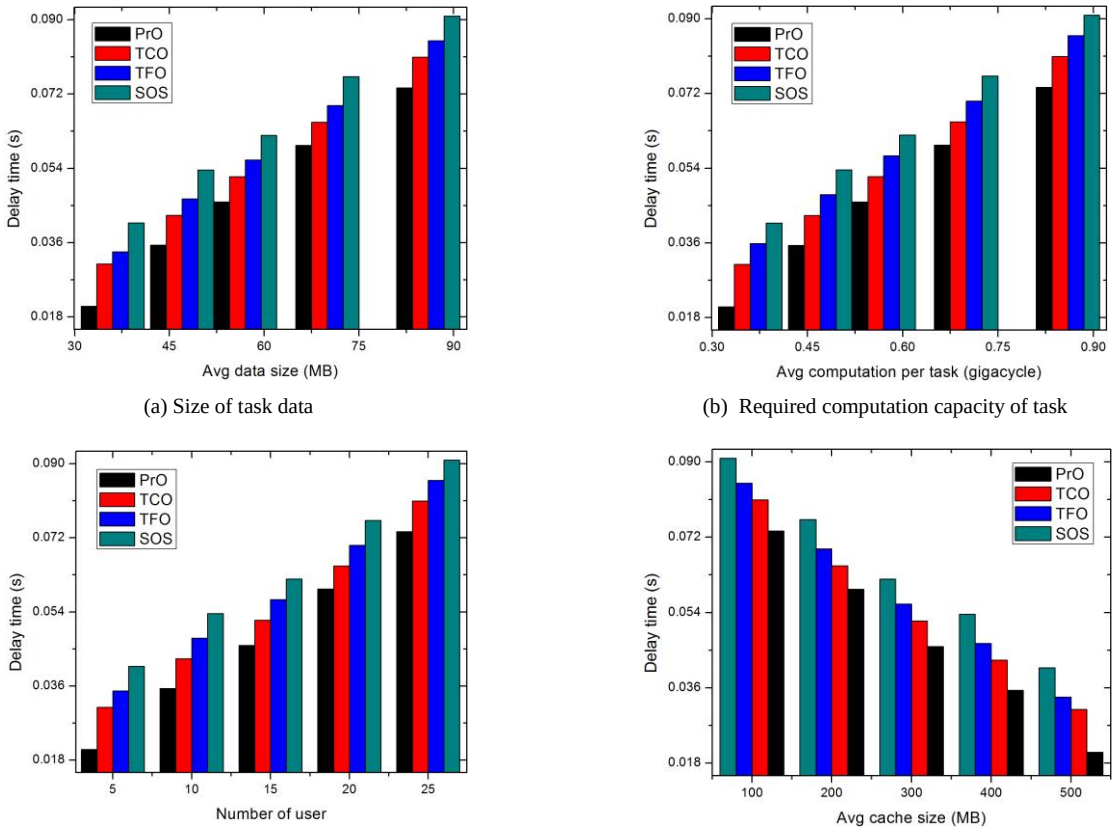
(c) Number of mobile users

(d) Edge cloud cache size

Fig. 4. Impacts of energy consumption on performances over the studied systems. The initial setting is p = 25, C^S = 500MB, Cr_q (using normal distribution) average of 0.6 gigacycles per task, S_a (using uniform distribution) average of 60 MB.

Impact of PrO scheme on energy consumption: In illustration of performance evaluation in fig. 4, it is clearly seen that the proposed PrO scheme reduces more energy cost than the FTO, SOS and TCO schemes. In fig. 4(a), it also shows that bigger the data size, the higher the energy cost. In fig. 4(b), it can be seen that required computation capacity almost has the same impact as the task data size. In fig. 4(c), it shows the impact of number of user on the energy consumption. In fig. 4(d), it shows the impact of edge cloud cache size on energy cost, the larger the cache size, the lower the energy consumption. In fig. 4, it is clear that our proposed priority-based task offloading and

caching scheme (PrO) is more energy efficient than any other schemes proposed earlier.



(a) Size of task data

(b) Required computation capacity of task

(c) Number of mobile users

(d) Edge cloud cache size

Fig. 5. Impacts of delay time on performances over the studied systems. The initial setting is p = 25, C^S = 500MB, Cr_q (using normal distribution) average of 0.6 gigacycles per task, S_q (using uniform distribution) average of 60 MB.

Impact of PrO scheme on delay time: In illustration of performance evaluation in fig. 5, it is clearly seen that the proposed PrO scheme reduces more delay time than the FTO, SOS and TCO schemes. Although, in this paper, the queuing time for each computing task is considered. In fig. 5(a), it also shows that larger the data size, the higher the delay time. In fig. 5(b), it can be seen that required computation capacity almost has the same impact as the task data size. In fig. 5(c), it shows the impact of number of user on the delay time. In fig. 5(d), it shows the impact of edge cloud cache size on delay time, the greater the cache size the lower the delay time. In fig. 5, it is clear that our proposed priority-based task offloading and caching scheme (PrO) is more delay efficient than any other schemes proposed previously.

VI. Conclusion

In our paper, a priority-based task offloading and caching scheme (PrO) is proposed for mobile edge computing. Our main target is to prioritize the computing task in a certain way that can reduce energy cost and delay time efficiently. The proposed solution helps us to decide which task to cache first and which or how much task to offload to the edge cloud. The queuing time is also considered along with transmission time and processing time to calculate the total execution time of a task for MEC.

The task with the highest priority is decided to compute locally to achieve the maximum user satisfaction. For that, a weighted priority is proposed using four most concerned aspects related to the task (task popularity, data size, deadline and computing resource). The joint optimization problem (task offloading and caching) is considered as mixed integer nonlinear programming. Here, sufficient algorithmic steps is provided to solve this joint optimization problem. In comparison to other schemes, the simulation result of our proposed scheme is more user efficient (less energy cost and delay time). For future work, there are some potential directions to extend the recent work. Joint user scheduling can be a potential direction for further research. Proper resource allocation for the task can gratify the low latency requirement and can reduce more energy cost. And finally, cooperative computation among close edge clouds (multi-edge cloud) can improve the performance of priority-based task offloading and caching scheme (PrO) for MEC.

References

[1] Yixue Hao, Min Chen, Long Hu, M Shamim Hossain, and Ahmed Ghoneim. Energy efficient task caching and offloading for mobile edge computing. IEEE Access, 6:11365–11373, 2018. DOI: 10.1109/ACCESS.2018.2805798.

[2] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim. Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Transactions on Wireless Communications, 16(3):1397–1411, 2017. DOI: 10/1109/TWC.2016.2633522.
[3] Ridhi Sharma et al. Computation offloading in mobile cloud computing. International Journal of Current Trends in Science and Technology, 7(12):20501–20510, 2017.
[4] Yonggang Wen, Weiwen Zhang, and Haiyun Luo. Energy-optimal mobile application execution: Taming resource-poor SMDs with cloud clones. In INFOCOM, 2012 Proceedings IEEE, pages 2716–2720. IEEE, 2012.
[5] Min Chen, Yixue Hao, Meikang Qiu, Jeungeun Song, Di Wu, and Iztok Humar. Mobility-aware caching and computation offloading in 5g ultradense cellular networks. Sensors, 16(7):974, 2016.
[6] Xiuhua Li, Xiaofei Wang, Keqiu Li, and Victor CM Leung. Caas: Caching as a service for 5g networks. IEEE Access, 5:5982–5993, 2017. DOI: 10.1109/ACCESS.2017.2689678.
[7] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking, (5):2795–2808, 2016.
[8] Anupam Bonkra and UG Student. Survey on computational offloading in mobile cloud computing environment. International Journal of Engineering Science, 11893, 2017.
[9] Neil Williams, Kheng Cher Yeo, and Sami Azam. Cost effective analysis of web based transmission. In Computational Intelligence for Communication Systems and Networks (CIComms), 2013 IEEE Symposium on, pages 72–78. IEEE, 2013.
[10] Fernaz Narin Nur and Nazmun Nessa Moon. Health care system based on cloud computing. Asian Transactions on Computers, 2(5):9–11, 2012.
[11] Antti P Miettinen and Jukka K Nurminen. Energy efficiency of mobile clients in cloud computing. HotCloud, 10:4–4, 2010.
[12] Ke Zhang, Yuming Mao, Supeng Leng, Quanxin Zhao, Longjiang Li, Xin Peng, Li Pan, Sabita Maharjan, and Yan Zhang. Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. IEEE access, 4:5896–5907, 2016. DOI: 10.1109/ACCESS.2016.2597169
[13] Cloud simulator cloudsim. http://code.google.com/p/cloudsim. Accessed: 2018-07-30.
[14] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. Femtocaching: Wireless content delivery through distributed caching helpers. IEEE Transactions on Information Theory, 59(12):8402–8413, 2013.
[15] Ying Cui, Wen He, Chun Ni, Chengjun Guo and Zhi Liu. Energy-Efficient Resource Allocation for Cache-Assisted Mobile Edge Computing.arxiv.org/abs/1708.04813v1.IEEE. DOI:10.1109/LCN.2017.112.

Saiful Islam attained his B.Sc. degree in Computer Science and Engineering from Daffodil International University, Bangladesh. His research interests in Cloud Computing, Cloud Security and Computer Networking.

Dr. Fernaz Narin Nur received her B.Sc. (Hons) from the Department of Computer Science and Engineering, Jahangirnagar University, Bangladesh in 2008 and M.S. from Institute of Information Technology, University of Dhaka, Bangladesh in 2010. She obtained her Ph.D. degree from the Department of Computer Science and Engineering, University of Dhaka, Bangladesh. She is now working as an Assistant Professor in the department of CSE of Daffodil International University, Bangladesh. She is a member of GNR (Green Networking Research group), IEEE, Bangladesh Women in IT. Her research interests include Wireless Sensor Network, Directional Wireless Sensor Network, Ad Hoc Networks, MAC Protocols, Performance Analysis, etc.

Nazmun Nessa Moon is a PhD student in the Department of CSE, BUET, Bangladesh. She received her B.Sc. (Hons) from the Department of Computer Science and Engineering, RUET in 2004 and MSc from Institute of Information Communication Technology, BUET in 2012. Her research interests include Graph Drawing and Algorithm, Graph Theory, Bioinformatics, Internet of Things. She is working in Daffodil International University since 2012.

Asif Karim is a PhD researcher at Charles Darwin University, Australia and lives in the port city of Darwin. His research interest includes Machine Intelligence and Cryptographic Communication. He is currently working towards the development of a robust and advanced email filtering system primarily using Machine Learning algorithms. Asif has considerable industry experience in IT, primarily in the field of Software Engineering.

Dr. Sami Azam is a leading researcher and lecturer at the college of Engineering and IT of Charles Darwin University, Australia. He is actively involved in the research fields relating to Computer Vision, Signal Processing, Artificial Intelligence and Biomedical Engineering. Dr. Azam has number of publications in peer reviewed journals and international conference proceedings.

Dr. Bharanidharan Shanmugam is a research intensive lecturer at the college of Engineering and IT of Charles Darwin University, Australia. He has a large number of publication in several different journals and conference proceedings. Dr. Shanmugam's research interest mainly revolves around the field of Cybersecurity.