Collaborative Cache Allocation and Computation Offloading in Mobile Edge Computing

Anselme Ndikumana, Saeed Ullah, Tuan LeAnh, Nguyen H. Tran, and Choong Seon Hong*

Department of Computer Science and Engineering, Kyung Hee University, Rep. of Korea
{anselme, saeed, latuan, nguyenth, cshong}@khu.ac.kr

***Abstract*— Mobile data traffic is increasing astronomically. This enormous extent was not only caused by the increasing in the number of mobile devices, but also the growing number of applications running on clouds. Most of these mobile devices have limited resources for computing, they are relying on cloud computing, which is inadequate to realize millisecond-scale latency in the 5G network. To deal with this issue, Mobile Edge Computing (MEC) has been introduced to supplement cloud computing by pushing Computing, Caching, Communication, and Control (4C) to the edges. However, the proposed MEC server at each base station is not enough to deal with 4C, when it operates independently, and without any collaboration with other MEC servers. This results in increasing the delay and making backhaul to continue suffering from huge data exchange between end-users and remote clouds. To address this challenge, we propose collaborative cache allocation and computation offloading, where the MEC servers collaborate for executing computation tasks and data caching. We formulate an optimization problem that aims at maximizing the resource utilization. The simulation results show that our proposal is easy to be implemented in production network.**

***Index Terms*—Collaborative Caching, Computation Offloading, Mobile Edge Computing, 5G network**

I. Introduction

Mobile data traffic is continuing to increase rapidly, and this increase is expected to continue in the coming years [1]. Due to this increase, the computation and data exchange between end-user devices and clouds are also increasing astronomically, where cloud data center traffic will increase from 3.9 ZB in 2015 to 14.1 ZB by 2020 [2].

To deal with both mobile data and cloud data traffic increase, Mobile Edge Computing (MEC) has been introduced to supplement cloud computing, where MEC servers are deployed at the Base Stations (BSs) for executing delay sensitive and context aware applications in close proximity to the end-users [3]. However, deploying MEC at the base station is not enough to deal with 4C and reduce significantly data exchange between end-users and remote clouds, when it operates independently, i.e., without any collaboration with other MEC servers. Therefore, to address this challenge, we propose collaborative cache allocation and computation offloading in MEC.

Our key contributions in this paper are summarized as follows:

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (B0190-16-2017, Resilient/Fault-Tolerant Autonomic Networking Based on Physicality, Relationship and Service Semantic of IoT Devices) *Dr. CS Hong is the corresponding author.

- We propose caching and computational resources allocation in MEC that satisfy multiple service requesters based on their demands and payments.
- We propose collaboration among MEC servers, where the MEC servers collaborate for executing computation tasks and data caching.
- We formulate an optimization problem that aims at maximizing the resource utilization.

As related work, in [4], the authors highlighted that caching spaces at edge nodes are usually small, which potentially results in low hit ratio. To overcome this issue, the authors highlighted the need of having inter-eNBs cooperative caching that allows low latency content delivery. In [5], the authors proposed a collaborative joint caching and processing strategy for on-demand video streaming in MEC networks. They formulate the collaborative joint caching and processing problem as an Integer Linear Program (ILP) that minimizes the backhaul network cost. In [6], the authors proposed a novel Cooperative Hierarchical Caching (CHC) framework for Cloud Radio Access Network (C-RAN), where the contents are jointly cached at the Base Band Unit (BBU) and at the Radio Remote Heads (RRHs) in order to improve cache hit ratio, decrease content access latency, and reduce backhaul traffic load.

The difference between our proposal with the above related works is that the caching and computation resources at MEC server are allocated to multiple service requesters based on their demands and payments.

The rest of the paper is organized as follows, Section II discusses in details our system model. Section III presents our collaborative cache allocation and computation offloading in MEC, while Section IV presents our performance evaluation. We conclude the paper and give future directions in Section V.

II. System Model

MEC network: In our system model depicted in Fig. 1, we consider MEC network of M MEC servers, where $\mathcal{M}$ is the set of MEC servers. MEC servers are connected via backhaul links. Each MEC server $i \in \mathcal{M}$ has both caching and computational resources that are divisible.

Service Requesters: We consider $\mathcal{K}$ as a set of service requesters, where each service requester $k \in \mathcal{K}$ is connected to its nearest BS referred as home BS based on its signal strength. Furthermore, each MEC server $i \in \mathcal{M}$ has resources which can be shared to multiple service requesters based on their demands as bids and payments, where z_{ki} is the computation request, while d_{ki} is caching request from service requester $k \in \mathcal{K}$.

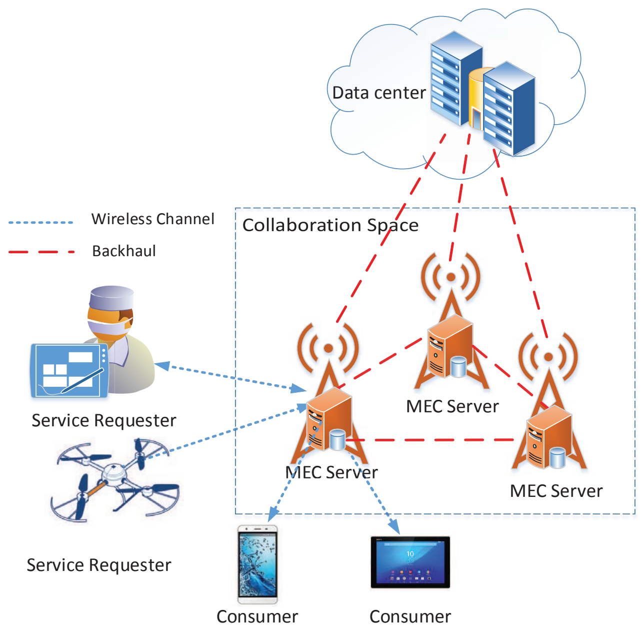

Figure 1: System model

In Fig. 1, we give two scenarios as examples from different domains that are good candidates for being executed at MEC servers: (i) drones can be used in professional sport activities to capture the sport game actions [7] and send live stream videos to its nearest MEC server $i \in \mathcal{M}$, in which MEC server performs live stream caching and processing. This heps the MEC server to satisfy consumers demands for variable bitrates, based on network condition and device capabilities. (ii) In medical imaging, instead of sending computation task to the remote cloud [8], medical imaging tasks can be sent to its nearest MEC server $i \in \mathcal{M}$ for being processed, and then MC server returns result to the service requester $k \in \mathcal{K}$.

Collaboration Space: We group all MEC servers under resources sharing at F hop count distance into a collaboration space. The network administrator can configure the value of F, i.e., the number of hops that MEC server $i \in \mathcal{M}$ needs to exchange resources with other MEC server(s). Furthermore, we consider that each MEC server $i \in \mathcal{M}$ has Resources Allocation Table (RAT) for recording available resources of other MEC server(s) located in the same collaboration space, where MEC servers exchange RAT updates on regular interval of time Δt.

We consider that the MEC servers in collaboration space belong to the same Mobile Network Operator (MNO), in which the MNO implements total cache storage C at the cost of $Q(C)$ and computation resources P at the cost $Q(P)$. Therefore, the total cache storage pool for MNO is expressed as $C = \sum_{i=1}^{M} C_i$, while the computation pool for MNO is expressed as $P = \sum_{i=1}^{M} P_i$. Furthermore, the total cost for both computation and caching implementation can be described as follows:

$$Q(C, P) = Q(C) + Q(P). \tag{1}$$

Once caching and computation resources are implemented, the MNO makes resources available to K service requesters, where $K \geq 2$. Let us consider y_i^k as a decision variable, which indicates whether or not the cache space is available at MEC server $i \in \mathcal{M}$. x_i^k indicates whether or not the computation resources is available at MEC server $i \in \mathcal{M}$. y_i^k/ $x_i^k = 1$ when the requested resources are available at MEC server $i \in \mathcal{M}$, and 0 otherwise. Let us denote y_j^k as a decision variable, which indicates whether cache space is available at any MEC server $j \in \mathcal{M}$ or not. Decision variable x_j^k indicates whether or not the computation resources is available at any other MEC server $j \in \mathcal{M}$ in collaboration space. MEC server $j \in \mathcal{M}$ and MEC server $i \in \mathcal{M}$ belongs to the same collaboration space. In case there are no available resources in neighboring MEC servers. MEC server $i \in \mathcal{M}$ forwards request to the remote cloud (data center) through the use of the backhaul.

III. Collaborative Cache Allocation and Computation Offloading

In this section, we discuss in details our collaborative cache allocation and computation offloading in MEC, where each service requester $k \in \mathcal{K}$ receives resources based on its demand and payment.

A. Resources Allocation

Each resource implemented at each MEC server $i \in \mathcal{M}$ is divisible, and can be used by multiple service requesters. Furthermore, in our analysis, unless otherwise stated, we will refer to MEC server $i \in \mathcal{M}$ and MEC server $j \in \mathcal{M}$ interchangeably, where the resource demands which are satisfied at server $i \in \mathcal{M}$ can be satisfied by any other MEC server $j \in \mathcal{M}$.

For getting resources, each service requester $k \in \mathcal{K}$ is asked to submit bid b_{ik} as demand, then MNO maps bid into the resource allocation and a payment for each service requester $k \in \mathcal{K}$ has to pay. Furthermore, for helping the service requesters to prepare their bids, MNO announces the available resources at each MEC server $i \in \mathcal{M}$ to service requesters, and the sum of the bids/demands placed on each MEC server $i \in \mathcal{M}$. However, the MNO does not reveal the bid of one service requester to another.

To allocate resources to multiple service requesters, let us denote r_{ki} as resource allocation, where $r_{ki}(c_{ki}, p_{ki})$ represents the resources (c_{ki} for caching and p_{ki} for computing) allocated to service requester k at MEC server $i \in \mathcal{M}$. Let us consider that each service requester k submits nonnegative demand as a bid $b_{ki}(d_{ki}, z_{ki})$ to its home BS (MEC server $i \in \mathcal{M}$), where $b_{ik} \geq 0$. We denote d_{ki} as bid for data caching, and z_{ki} as bid for computation offloading submitted by service requester $k \in \mathcal{K}$ at MEC server $i \in \mathcal{M}$. The total bids submitted at each MEC server $i \in \mathcal{M}$ for caching becomes $\sum_{k:i\in\mathcal{M}} d_{ki}$, while the total bids for computing becomes $\sum_{k:i\in\mathcal{M}} z_{ki}$. The MNO allocates resources based on weighted proportional allocation [9], where each service requester $k \in \mathcal{K}$ receives the fraction of resources at MEC server $i \in \mathcal{M}$ as follows:

$$c_{ki} = C_i \frac{d_{ki}}{\sum_{k:i\in\mathcal{M}} d_{ki}}, \tag{2}$$

$$p_{ki} = P_i \frac{z_{ki}}{\sum_{k:i\in\mathcal{M}} z_{ki}}, \tag{3}$$

where C_i is the cache space available at MEC server $i \in \mathcal{M}$, while P_i available computation resource at MEC server $i \in \mathcal{M}$.

For $\sum_{k:i\in\mathcal{M}} d_{ki} = 0$, the cache space at each MEC server $i \in \mathcal{M}$ is free of content, i.e., it does not allocated to any service requester, while $\sum_{k:i\in\mathcal{M}} z_{ki} = 0$ means that MEC server $i \in \mathcal{M}$ does not have any task to compute.

We consider that the MNO does not have infinite resources. MNO has cache space C_i constraint, and computation resource P_i constraint at each MEC server $i \in \mathcal{M}$, where the total cache allocation $\sum_{k:i\in\mathcal{M}} c_{ki}$ and total computation allocation $\sum_{k:i\in\mathcal{M}} p_{ik}$ at MEC server $i \in \mathcal{M}$ must be less than or equal the total cache space C_i and computation resource P_i.

$$\sum_{k:i\in\mathcal{M}} y_i^k c_{ki} \leq C_i, \tag{4}$$

$$\sum_{k:i\in\mathcal{M}} x_i^k p_{ki} \leq P_i. \tag{5}$$

Each service requester $k \in \mathcal{K}$ is associated with utility function $U_k(c_{ki}, p_{ki})$ of the fraction of the resources r_{ki} receives from MEC server $i \in \mathcal{M}$. Each service requester $k \in \mathcal{K}$ gives weight its resources, where we denote $w_k(w_{ki}, v_{ki})$ as a vector of the weights. The utility function now becomes linear function:

$$U_k(c_{ki}, p_{ki}) =< w_{ki}c_{ki}, v_{ki}p_{ki} >, \tag{6}$$

where $0 \leq w_{ki} \leq 1$ and $0 \leq v_{ki} \leq 1$ are the private preferences of each service requester $k \in \mathcal{K}$ for computation and cache resources assigned to him at MEC server $i \in \mathcal{M}$.

B. Pricing Framework

From the resource r_{ki} allocated to service requester $k \in \mathcal{K}$ at MEC server $i \in \mathcal{M}$ in the above subsection III-A, each service requester $k \in \mathcal{K}$ needs to contribute to $Q(C, P)$ through payment, where the resource allocation function is known by service requesters in advance before submitting their bids.

We consider weighted payment, in which the MNO assigns the weight θ_{ki} to each payment needs to be paid by service requester $k \in \mathcal{K}$ based on MEC server $i \in \mathcal{M}$ location. MNO gives discount to each service requester $k \in \mathcal{K}$ for the resources allocated to him at MEC server $j \in \mathcal{M}$, which is different than its nearest MEC server $i \in \mathcal{M}$ attached to its home BS. Thus, the delay from MEC server $i \in \mathcal{M}$ to the service requester $k \in \mathcal{K}$ is less than the delay from MEC server $j \in \mathcal{M}$ to the service requester $k \in \mathcal{K}$. In order to adopt this discount in our proposal, a robust pricing framework is needed that consider the MEC server location.

We consider q_{ki} as the payment that the service requester $k \in \mathcal{K}$ has to pay to its MNO for the resource(s) allocated to him at MEC server $i \in \mathcal{M}$. We consider that the resources of one MEC server $i \in \mathcal{M}$ are not enough to cover all the resource demands. Each MEC server $i \in \mathcal{M}$ collaborates with its neighboring MEC servers at F hop count distance.

Based on the distance between service requesters and MEC servers, the resources have different values from both MNO and service requesters. Each service requester $k \in \mathcal{K}$ gives w_k to each MEC server, and it is willing to get more resources at its nearest MEC server $i \in \mathcal{M}$, so that end-to-end delay can be reduced. On the other side, we consider that the MNO sets also the weight θki based distance between service requesters and MEC server $i \in \mathcal{M}$, where $\theta_{ki} = \frac{1}{|F|}$ is the weight that needs to be applied to the payment q_{ki} that each service requester $k \in \mathcal{K}$ has to pay to MNO for the resources allocated to him at MEC server $i \in \mathcal{M}$. The q_{ki} is defined as follows:

$$q_{ki}(d_{ki}, z_{ki}) = \theta_{ki}c_{ki}d_{ki} + \theta_{ki}p_{ki}z_{ki}. \tag{7}$$

The payoff of each service requester $k \in \mathcal{K}$ is expressed as follows:

$$R_k = U_k(c_{ki}, p_{ki}) - q_{ki}(d_{ki}, z_{ki}), \tag{8}$$

while the payoff of MNO is equal to:

$$S(d_{ki}, z_{ki}) = \sum_{k=1}^{K} q_{ki}(d_{ki}, z_{ki}) \leq Q_i(C_i, P_i), \tag{9}$$

where $Q_i(C_i, P_i)$ is the implementation cost of total cache space C_i and computation resource P_i at MEC server $i \in \mathcal{M}$.

We consider each service requester $k \in \mathcal{K}$ has a budget constraint B_k, where its bid b_{ki} must be less than or equal to its budget B_i. From this perspective, each service requester $k \in \mathcal{K}$ chooses to offload his task to MEC server though solving the following optimization problem:

$$\max_{y_i^k, x_i^k} \quad y_i^k x_i^k U_k(c_{ki}, p_{ki}) \tag{10}$$

$$\text{s.t.} \quad \sum_{k:i\in\mathcal{M}} y_i^k c_{ki} \leq C_i, \ \forall i \in \mathcal{M}, \ k \in \mathcal{K} \tag{11}$$

$$\sum_{k:i\in\mathcal{M}} x_i^k p_{ki} \leq P_i, \ \forall i \in \mathcal{M}, \ k \in \mathcal{K} \tag{12}$$

$$y_i^k x_i^k b_{ki} \leq B_k, \ \forall i \in \mathcal{M}, \ j \in \mathcal{K} \tag{13}$$

$$y_i^k, x_i^k \in \{0, 1\}, \ \forall i \in \mathcal{M}, \ j \in \mathcal{K}. \tag{14}$$

Algorithm 1 Algorithm for cache allocation and computation offloading in MEC

```
1:  Input: $C_i$, $C_j$, $P_i$, $P_j$, $\forall i, j \in \mathcal{M}$,
2:  Initialize: $c_{ki} = 0$, $p_{ki} = 0$, $c_{kj} = 0$, $p_{kj} = 0$, $\forall i, j \in \mathcal{M}$, $k \in \mathcal{K}$
3:  for each request for $< d_{ki}, z_{ki} >$ arriving at MEC server $i \in \mathcal{M}$ do
4:    Allocate $< c_{ki}, p_{ki} >$ at MEC server $i \in \mathcal{M}$
5:    Send RAT update
6:    if $\sum_{k:i\in\mathcal{M}} y_i^k c_{ki} = C_i$ and $\sum_{k:i\in\mathcal{M}} x_i^k p_{ki} = P_i$ then
7:      Check RAT
8:      if $\sum_{j:k\in\mathcal{K}} y_j^k c_{kj} \leq C_j$ and $\sum_{j:k\in\mathcal{K}} x_j^k p_{kj} \leq P_j$ then
9:        Forward request to MEC server $j \in \mathcal{M}$
10:     else
11:       Forward request to remote cloud
12:     end if
13:   end if
14: end for
15: for each request for $< d_{kj}, z_{kj} >$ arriving at MEC server $j \in \mathcal{M}$ do
16:   Allocate $< c_{kj}, p_{kj} >$ at MEC server $j \in \mathcal{M}$
17:   Send RAT update
18: end for
```

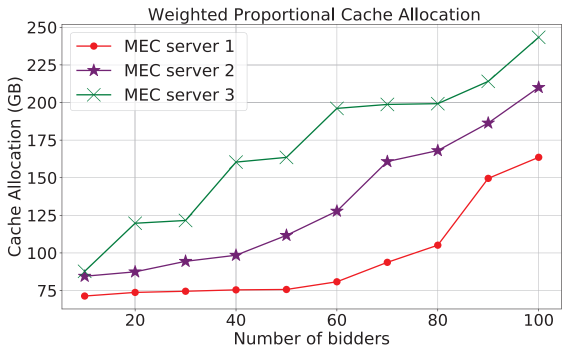

Figure 2: Weighted proportional cache allocation

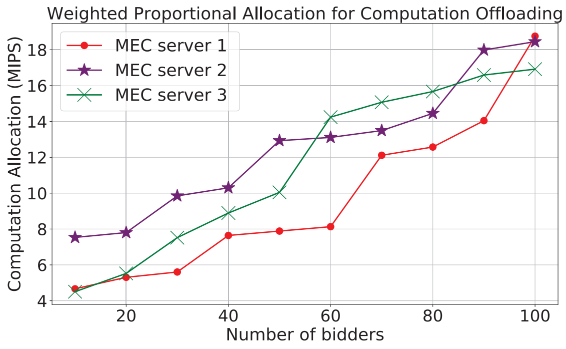

Figure 3: Weighted proportional allocation for computation offloading

The problem (10) is NP-hard, where its NP-hardness can be approved by an easy reduction in formulated conjunctive normal form [10]. In order to reduce the computational complexity of our Integer Linear Programing (ILP) problem (10), we propose an algorithm 1 for coupled computation and data caching in MEC.

IV. Performance Evaluation

In this section, we present the performance evaluation of our proposal. During the evaluation, we use numerical analysis, where Julia language [11] is used.

A. Experimental Setup

In our experimental setup, we consider the number of servers to be $M = 10$, while the number of bidders/ service requesters are from $K = 10$ to $K = 100$. In subsequent analysis, unless otherwise stated, we will refer to service requesters and bidders interchangeably. The arrival rate at each MEC server is from 10 to 100. Furthermore, in each demand, d_k is generated randomly and uniformly distributed from \$2 to \$7 per 1GB, while z_k is generated randomly and uniformly distributed from 2\$ to 10\$ per 1 Million of Instructions Per Second (MIPS). The cache storages of MNO are randomly and uniformly distributed from 1000 to 10000 GB, while computation capacities are randomly and uniformly distributed in the range from 100 to 1000 MIPS per each MEC server.

B. Simulation Results

In our simulation results, we use 10 MEC servers. All MEC servers use the same resource allocation technique (weighted proportional allocation). Therefore, for making our presentation of the results easy to visualize, we present the results of 3 MEC servers.

Fig. 2 shows weighted proportional cache allocation in term GB. The cache storages are allocated proportional based on demands and payments, where the bidder $k \in \mathcal{K}$ who demands and pays more, gets more resources. This results in making demands and resource allocation graphs to follow the same slopes.

Fig. 3 presents weighted proportional allocation for computation offloading in terms of MIPS. The computation resources are allocated proportional based on demand and payment, subject to the processing capacity constraint.

V. Conclusion and Future directions

In the MEC, the proposed MEC server in the literatures at each base station is not enough to deal with 4C, and reduce significantly data exchange between end-users and remote clouds, when it operates independently, and without any collaboration with other MEC servers. Therefore, to address this challenge, in this paper, we proposed collaborative cache allocation and computation offloading, where the MEC servers collaborate for executing computation tasks and data caching. In our proposal, caching and computational resources are allocated to many service requesters based on their demands and payment, where MNO allocates resources based on weighted proportional allocation. We formulate an optimization problem that aims at maximizing the resource utilization. In the future, we aim to extend our proposal with more simulations and analysis.

References

[1] Cisco, “Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper,” http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, [Online; accessed May 21, 2017].

[2] TechRepublic, “Cloud traffic to jump 262% by 2020,” http://www.techrepublic.com/article/cloud-traffic-to-jump-262-by-2020-according-to-cisco-global-cloud-index, [Online; accessed May 21, 2017].

[3] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, “Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges,” *arXiv preprint arXiv:1612.03184*, 2016.

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, “Cache in the air: exploiting content caching and delivery techniques for 5g systems,” *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[5] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, “Collaborative multi-bitrate video caching and processing in mobile-edge computing networks,” *arXiv preprint arXiv:1612.01436*, 2016.

[6] T. X. Tran, A. Hajisami, and D. Pompili, “Cooperative hierarchical caching in 5g cloud radio access networks (c-rans),” *arXiv preprint arXiv:1602.02178*, 2016.

[7] chicagotribune, “Drones used to capture high school football action,” http://www.chicagotribune.com/suburbs/elgin-courier-news/news/ct-high-school-football-drone-met-20141024-story.html, [Online; accessed May 21, 2017].

[8] G. C. Kagadis, C. Kloukinas, K. Moore, J. Philbin, P. Papadimitroulas, C. Alexakos, P. G. Nagy, D. Visvikis, and W. R. Hendee, “Cloud computing in medical imaging,” *Medical physics*, vol. 40, no. 7, 2013.

[9] T. Nguyen and M. Vojnovic, “Weighted proportional allocation,” in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. ACM, 2011, pp. 173–184.

[10] V. Gurvich and L. Khachiyan, “On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions,” *Discrete Applied Mathematics*, vol. 96, pp. 363–373, 1999.

[11] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A fresh approach to numerical computing,” *arXiv preprint arXiv:1411.1607*, 2014.