Dynamic Clustering for Low-Delay Delivery of Video Content Cached in MEC Servers

Ali Doostmohammadi, Mohammad Reza Khayyambashi, Naser Movahedinia, *Member, IEEE*, and Zdenek Becvar, *Senior Member, IEEE*

***Abstract*—For the purpose of video caching and low-delay video delivery to the end-users, multiaccess edge computing (MEC) servers are commonly grouped into clusters to efficiently exploit the limited storage resources of the MEC servers. In this article, we first introduce a methodology for analysis of the video delivery delay using the queuing theory. Our analysis shows that the video delivery delay is mainly affected by the arrival rate of the video requests. Furthermore, we show a tradeoff between the ratio of videos found in the MEC servers within the same cluster and the transmission delay of video contents. To reduce the video delivery delay, we propose a dynamic MEC server clustering (DyMECC) algorithm that determines the cluster size at each time interval by solving analytically derived equations considering the actual arrival rate of video requests. It also acts as a congestion avoidance mechanism for the communication interfaces among the MEC servers. Via simulations, we show that the DyMECC reduces the video delivery delay about 15% in light load conditions and by more than five times in heavy load conditions compared to state-of-the-art works while also reduces the load of the communication interfaces among the MEC servers by more than 45%.**

***Index Terms*—6G, clustering, cooperative caching, multiaccess edge computing (MEC), video, Xn interface.**

I. Introduction

According to the Ericsson mobility report [1], the traffic in mobile networks should be 110 exabytes per month by 2023. Out of this volume of data, 45% belongs to video [2], [3]. Future mobile networks should deal with this tremendous video traffic. Although 6G visions promise high bit rates and ultra-low latency [4], [5], there is a huge difference between the transmission delay of video contents in 6G radio access network and the delay of fetching the videos from the original content providers via a core network and the Internet [6]. A low-delay video delivery can be facilitated by MEC servers that provide the storage resources for caching the popular video contents close to the users, i.e., at the edge of the mobile network. Thus, MEC-assisted video content delivery has been recently a dominant paradigm toward reducing the video delivery delay, as shown, e.g., in [7], [8], [9], and [10]. Caching the video contents at the storage of MEC servers not only reduces the content delivery delay, but also reduces the backhaul traffic load via not fetching the video contents from the far-away servers of the original video content provider [11], [12].

To further improve caching performance at the edges, video conversion and recommendation have been introduced. These techniques work to produce lower quality versions of videos or provide alternative similar videos when requested videos are not available [13], [14]. In [15], an efficient algorithm was introduced to jointly utilize the video conversion and video recommendation for reducing the video delivery delay.

As the quality of videos (e.g., resolution, video bit rate, multiview 3-D videos) increases over time, more storage resources are required in the MEC servers to keep the video contents close to the users and to reduce the video delivery delay as well as the backhaul load. To this end, an effective caching mechanism with an adaptive view selection is introduced in [16]. In [17], a MEC-assisted caching of multiview 3-D video with a dynamic view angle selection was introduced to reduce the required storage requirements by real-time synthesizing of the desired views. The authors presented a novel cache management problem called Adaptive View Selection and Cache Operation, and derive an optimal policy using the Markov decision process. Nevertheless, these works assume just single MEC server as a caching proxy; hence, a large storage space is still needed.

To avoid an increase in the cost of storage in the MEC servers, collaborative caching is adopted in [13], [18], and [19]. In the cooperative caching, several neighbouring base stations (BS) collocated with MEC servers share their cached contents. In [18], cache-enabled vehicles cooperated with base stations on sharing of the cached content. This idea was further evolved to a digital twin empowered content caching in vehicular edge networks in [20]. In hierarchical ultradense networks, the cooperative caching can be extended to the cooperation between micro BSs and macro BSs as well. For example, a collaborative caching scheme for virtual reality (VR) video delivery in MEC-enabled small-cell networks was proposed in [21] aiming to obtain low-delay VR video delivery.

One drawback of the cooperative caching is an existence of the duplicate video contents in adjacent MEC servers leading

Manuscript received 27 October 2022; revised 13 April 2023; accepted 28 May 2023. This work was supported in part by the Ministry of Education, Youth and Sport of the Czech Republic under Grant LTT20004, and in part by Czech Technical University in Prague under Grant SGS20/169/OHK3/3 T/13. *(Corresponding author: Mohammad Reza Khayyambashi.)*

Ali Doostmohammadi is with the Department of Computer Architecture, Faculty of Computer Engineering, University of Isfahan, Isfahan 73441, Iran, and also with the Faculty of Electrical Engineering, Czech Technical University in Prague, Prague 73441, Czech Republic (e-mail: a.doostmohammadi@eng.ui.ac.ir).

Mohammad Reza Khayyambashi and Naser Movahedinia are with the Department of Computer Architecture, Faculty of Computer Engineering, University of Isfahan, Isfahan 81746, Iran (e-mail: M.R.Khayyambashi@comp.ui.ac.ir; naserm@eng.ui.ac.ir).

Zdenek Becvar is with the Faculty of Electrical Engineering, Czech Technical University in Prague, Prague 16627, Czech Republic (e-mail: zdenek.becvar@fel.cvut.cz).

Digital Object Identifier 10.1109/JSYST.2023.3283965

to redundancy in video contents and inefficient usage of the caching resources in the MEC server [22]. To limit the redundancy and consequent requirements on over-dimensioning of the storage resources in the MEC servers, the MEC servers can be grouped into clusters as indicated in [22] and [23]. The MEC server clustering can be classified into two categories: 1) *user-centric clustering:* where the user accesses the BS via radio interface and receives the content directly from the accessed BS, as introduced, e.g., in [24], [25], [26], and [27]; and 2) *interconnection-based clustering:* where the MEC servers collocated with the BSs are grouped together to construct a coherent entity in terms of caching and the user accesses the serving BS, which requests the content from other BSs in the cluster via interfaces [22], [23], [28].

In the user-centric clustering of the MEC servers, the communication range of the users and the BSs limits the size of the cluster. In contrast, the interconnection-based clustering expands the domain of the cooperating MEC servers to a further distance in which the video contents can be exchanged through the interconnected MEC servers. Thus, in this article, we focus on the interconnection-based clustering and the communication interface between the MEC servers can be of any kind. We use Xn interface throughout the article as it is commonly used in 5G and beyond mobile networks.

A direct connection of the BSs via X2 interface for the purpose of the cooperative caching is introduced in [22], where the MEC servers were connected to each other to form a cluster. the clusters are fixed during the network setup and do not change during the system operation even if the rate of video requests varies over time. In [23], the MEC servers were connected to each other and logically grouped into the clusters based on the user density distribution and the geographical location of MEC servers. The authors introduced a collaborative caching splitting the storage resource of the MEC servers into local, intracluster, and intranetwork partitions aimed to minimize the average delay of content delivery. Nevertheless, the collaborative caching is not designed to operate in a heterogeneous and dynamic environment, and like in the previous work, the clustering scheme does not change over time. Moreover, the rate of the users' requests is not taken into account in the calculation of content delivery delay. In [28], the moving users were classified into different groups based on their velocity. Then, different MEC server clustering was defined for each user group to minimize the end-to-end delay of network services. Although a kind of MEC server clustering is introduced in this article, the authors focus on the reduction of the virtualized network functions migration among clusters and the dynamic nature of user demands is not considered.

Besides the abovementioned works on the fixed clustering of MEC servers, some papers target also dynamic video delivery path. For example, a dynamic routing for transmitting the videos among the interconnected MEC servers was proposed in [29] to offload the backhaul traffic to X2 interfaces. From the application focus perspective, this work focuses on cooperative caching and traffic offloading for video streaming services. From the technology approach perspective, this work proposes the use of content centric network and software defined network (SDN) to optimize network resource utilization. Besides this work does not consider the real-time dynamics of user requirements when optimizing network resource utilization, which could limit its effectiveness in scenarios with highly dynamic user requirements.

Also, an online caching policy was proposed for reducing the content delivery delay in content delivery networks in [30]. Although this algorithm is proven to be effective in the environments where popularity is highly dynamic, our proposed work differs from [30] in addressing dynamic aspects, since our proposal focuses on the dynamic nature of video request arrival rate, while [30] focused on the dynamic nature of the popularity of the multimedia data. Consequently, problem formulations as well as solutions in our manuscript and [30] are completely different.

In this article, we first analyze the video delivery delay using the queuing theory and, then, we propose the algorithm that dynamically adjusts the cluster size according to the arrival rate of video requests to minimize the video delivery delay. Our major contributions are summarized as follows.

1) We introduce an analytical framework for modeling the video delivery delay using the queueing theory. We take not only the waiting times into account, but also the variations in the service rate of the clusters, the arrival rate of video requests, and the cluster size.
2) We analyze the tradeoff between the probability of finding the video content in the cluster and the delivery delay of the video content within the cluster to obtain the cluster size that minimizes the video delivery delay.
3) We propose a dynamic MEC server clustering framework to minimize the video delivery delay in the presence of the variable arrival rate of video requests. The cluster size is determined based on the observed arrival rate of the video requests in each time interval to efficiently utilize the Xn interfaces.
4) We demonstrate that the proposed algorithm significantly reduces the video delivery delay, even more than five times in heavy load conditions, compared to state-of-the-art works. At the same time, the load of the communication interfaces among the MEC servers is reduced notably, by more than 45%, by the proposal.

The rest of this article is organized as follows. The system model is described in Section II. The analytical model for the video delivery delay and the tradeoff analysis is introduced in Section III. Then, Section IV presents the formulation of the optimization problem. Our proposed DyMECC is outlined in Section V. Section VI provides the performance evaluation. Finally, Section VII concludes the article.

II. System Model

In this section, we outline system model for the delivery of video content to users. Note that the notations used throughout this article are summarized in Table I.

We assume that $\mathbf{M}$, $\mathbf{U}$, $\mathbf{V}$ are the set of MEC servers, the set of users, and the set of videos, respectively. Each MEC server is collocated with one BS and each BS (and thus, each MEC server) is connected to the core network via the backhaul. Besides, each BS can directly communicate with its neighbors via Xn interface as illustrated in Fig. 1.

The MEC server m has a storage capacity of C_m. For each MEC server, a binary variable $\Delta_{m,v}$ shows if the vth video is cached in the mth MEC server ($\Delta_{m,v} = 1$) or not ($\Delta_{m,v} = 0$).

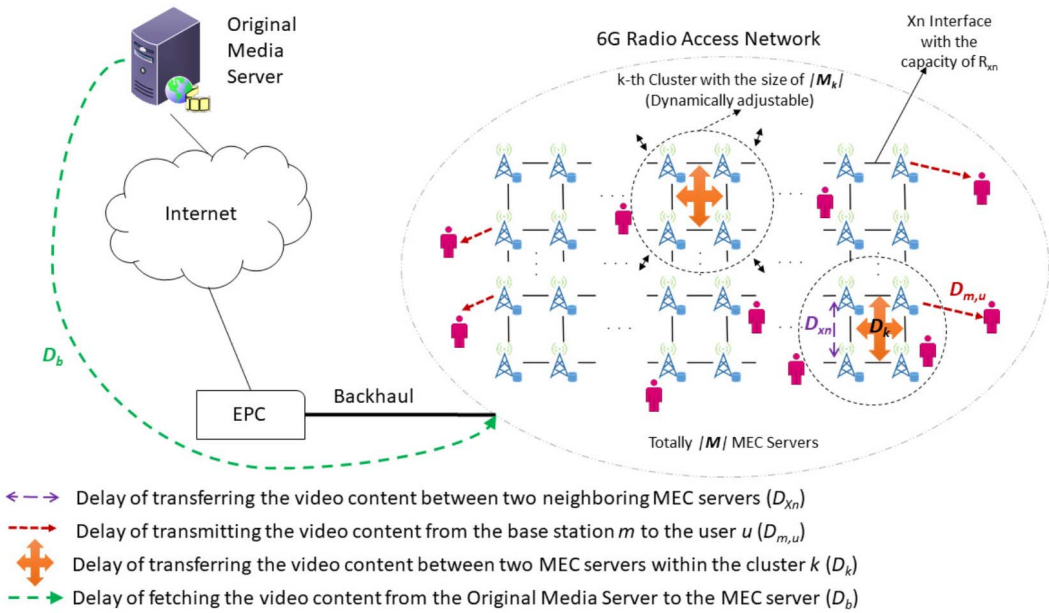

Fig. 1. System model for the delivery of video content to the users. D_{Xn} is applied when one MEC server transfers the video content to one of its neighboring MEC server and is a function of the video size and the capacity of Xn interface, whereas D_K is the delay of transmitting the video content between two arbitrary MEC servers within the cluster and is a function of the cluster size. In fact, D_k is applied when the requested video is delivered by the MEC server within the cluster k. D_b is applied when the requested video is not found in the cluster k and is fetched from the original media server. $D_{m,u}$ is always applied because the serving MEC server is the one that transmits the requested video to the user u.

TABLE I
NOTATION OF VARIABLES AND PARAMETERS USED IN THE MODEL

Symbol	Description
$\mathbf{M}, \mathbf{U}, \mathbf{V}$	Set of MEC servers, set of users, set of videos
K	Number of clusters
M_k	Set of MEC servers in cluster k
U_m	Set of users served by MEC server m
C_m	Storage capacity of MEC server m
S_ν	Size of video ν
λ_m	Arrival rate of video request to MEC server m
μ_m	Service rate of MEC server m
λ_k	Service rate of cluster k
R_{Xn}	Capacity of Xn interface
D_{Xn}	Transmission delay between two neighboring MEC server
$D_{m,u}$	Transmission delay from MEC server m to user u
D_k	Intra-cluster delay of cluster k
D_b	Transmission delay from media server to MEC server
$\alpha_{m,\nu}$	Probability of finding video ν in MEC server m
$\beta_{k,\nu}$	Probability of finding video ν in cluster k
$\rho_{\nu,u}$	Probability of requesting video ν by user u

The distribution of video popularity commonly follows Zipf distribution, as shown, e.g., in [31], so the video v is requested by the user u with the probability of

$$\rho_{\nu,u} = \left(\nu^{\gamma} \sum_{n=1}^{|\mathbf{V}|} n^{-\gamma} \right)^{-1} \tag{1}$$

where γ is the Zipf parameter indicating the popularity skewness.

When the MEC server m receives a request for the video v from the user u, the following three different cases may occur.

1) *Local cache-hit:* the MEC server m checks its cache. If the video v is found in its cache with the probability of $\alpha_{m,v}$, the MEC server m sends the requested video to the user directly via collocated BS. In this case, the edge transmission delay $D_{m,u}$ equals to the video transmission delay from the serving BS collocated with the MEC server m to the user u.
2) *Cluster-hit:* If the requested video is not found in the cache of the MEC server m, the MEC server m requests the video from other MEC servers in the same cluster k (i.e., the cluster to which the MEC server m belongs to). If one of the MEC servers within the cluster k finds the requested video in its cache, the video is sent to the MEC server m via the Xn interface connecting the MEC servers to each other. In this case, the intracluster delay D_k includes the video transmission delay between the MEC servers within the cluster.
3) *Fetching from the original media server:* If the requested video is not found in the cluster, the MEC server m fetches the requested video from the original media server via backhaul and evolved packet core network. In this case, the video delivery delay from the original media server to the MEC server m is represented by the backhaul delay D_b.

The cluster-hit probability is calculated based on the local cache-hit probability of individual members of the cluster. Precisely, the cluster-hit probability is the complement of the probability of not finding the requested video in any caches of the MEC servers within the cluster. Hence, the cluster-hit probability of finding the video v in the cluster k is defined as

$$\beta_{k,\nu} = 1 - \prod_{i=1}^{|\mathbf{M}_k|} (1 - \alpha_{i,\nu}) \tag{2}$$

where $|\mathbf{M}_k|$ is the number of MEC servers in the cluster k.

In this article, we assume a practical and realistic case, where there are many different video contents in the video library and the total size of the video library is huge in comparison with the cache size of MEC servers, i.e., $\sum_{v \in \mathbf{V}} S_v \gg C_m, \quad \forall m \in \boldsymbol{M}$,

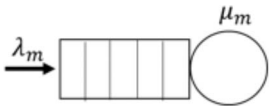

Fig. 2. MEC server as a queue.

Fig. 3. Cluster of MEC servers as a queue with a variable service rate.

where S_v is the size of the video v in the video library $\boldsymbol{V}$. Therefore, the local cache-hit probability of all MEC servers are very low and approximately the same, i.e., $\alpha_{m,v} = \alpha, \quad \forall m \in \boldsymbol{M}, \forall v \in \mathbf{V}$. Hence, the cluster-hit probability is solely a function of local cache-hit and the cluster size, i.e.,

$$\beta_{k,\nu} = \left(1 - (1-\alpha)^{|\mathbf{M}_k|}\right). \tag{3}$$

In conclusion, the system model represents the video content delivery to the end users through the interconnection network of MEC servers, taking into account a range of factors such as the storage capacity, the video size, the interface capacity, and various transmission delays. By examining different caching scenarios, including local cache-hits, cluster-hits, and fetching from the original media server, the model offers a realistic portrayal of how video requests are handled within a clustered MEC environment. Additionally, the model integrates the impact of video popularity based on a Zipf distribution and investigates the connection between local cache-hit probability, cluster-hit probability, and the various system parameters. This cooperative approach to the system model paves the way for a thorough analysis and evaluation of the proposed DyMECC algorithm, showcasing its effectiveness in reducing the video delivery delay compared to the fixed clustering schemes.

III. Analytical Model of Video Delivery Delay and Tradeoff Analysis

In this section, first, we introduce an analytical model for the video delivery delay. Then, we investigate the tradeoff between the cluster-hit ratio and the intracluster delay. We also analyze the effect of the backhaul delay on the cluster size.

A. Analytical Model

Since the capacity of the Xn interfaces interconnecting the MEC servers in mobile networks is finite, when the video content should be transmitted via the Xn interface that is transmitting another video content, the video content enters a queue for that Xn interface. Therefore, the arrival rate of the video requests affects the video delivery delay. Higher arrival rate of the video requests results in the higher queue waiting times and, consequently, a higher video delivery delay.

Using queuing theory, we model each MEC server as a queue of the video requests. The video requests arrive in MEC server m with the rate λ_m. When a new video request arrives, it is placed to the end of the queue (Fig. 2). The service rate μ_m of MEC server m is the rate at which MEC server m transmits the video content to its users $\mathbf{U}_m$.

We assume that the interarrival times of video requests and the service times of MEC servers (i.e., the time taken to transmit video content) are the Poisson point processes [23]. Therefore, the average transmission delay D_m of the video requested from the MEC server m to the user u is derived as

$$D_{m,u} = \frac{1}{\mu_m - \lambda_m} \tag{4}$$

where λ_m is the arrival rate of video requests on the MEC server m, and μ_m is the rate at which the MEC server m transmits the video contents to the $\mathbf{U}_m$ users over the radio resources with the bandwidth B, hence, the average transmission rate is formulated as

$$\mu_m = \frac{1}{|\mathbf{U}_m|} \sum_{u \in \mathbf{U}_m} B \times log_2 \frac{P^w L_{m,u}^{-\varepsilon}}{\sigma^2 + I_m} \tag{5}$$

where P^w is the transmission power, $L_{m,u}$ is the distance between the MEC server m and the user u, ε is the path loss exponent, σ^2 is the additive Gaussian noise power density, and I_m denotes the intercell interference.

Based on the model for individual MEC servers, each cluster of MEC servers is modeled also as a queue with an arrival rate of λ_k and with a variable service rate of μ_k (Fig. 3).

Then, based on the queuing theory, the intracluster video delivery delay is modeled as

$$D_k = \frac{1}{\mu_k - \lambda_k}. \tag{6}$$

The arrival rate λ_k of the kth cluster is an aggregation of the arrival rates of the MEC servers belonging to the kth cluster, i.e., $\sum_{m \in \mathbf{M}_k} \lambda_m$ multiplied by the probability of finding the requested video in the cluster, i.e., $\sum_{v \in \mathbf{V}} \rho_v \beta_{k,v}$. Thus, the arrival rate λ_k is defined as

$$\lambda_k = \sum_{m \in \mathbf{M}_k} \lambda_m \times \sum_{v \in \mathbf{V}} \rho_v \beta_{k,v}. \tag{7}$$

If we substitute $\beta_{k,v}$ with its definition in (3) and assuming the arrival rate of all MEC servers is the same ($\lambda_m = \lambda, \forall m \in \mathbf{M}$), we get

$$\lambda_k = (|\mathbf{M}_k| \times \lambda) \times \left(1 - (1-\alpha)^{|\mathbf{M}_k|}\right). \tag{8}$$

The service rate μ_k of the kth cluster is the inverse of the average transmission time of the video content with the size S within the cluster, where S is the average size of the videos in the video library, i.e., $S = \frac{1}{|\mathbf{V}|} \sum_{\nu \in \mathbf{V}} S_\nu$. The average transmission delay of the video content between two neighboring MEC servers directly connected to each other via Xn interface with the capacity of R_{Xn} is $D_{Xn} = \frac{S}{R_{Xn}}$. Let ξ_k be the average distance in terms of the number of MEC servers in the video delivery path between two MEC servers in the cluster k with $|\mathbf{M}_k|$ MEC servers. Then, we have

$$\mu_k = \frac{1}{D_{Xn} \times \xi_k}. \tag{9}$$

Using the delays in the video delivery described above, we can calculate the overall average video delivery delay perceived by the users served by the MEC server m belonging to the cluster

k as

$$D_m = \frac{1}{|\mathbf{U}_m||\mathbf{V}|} \sum_{u\in\mathbf{U}_m} \sum_{\nu\in\mathbf{V}} \rho_{v,u}[\alpha_{m,\nu}.D_{m,u} + (1-\alpha_{m,\nu})\beta_{k,v} \times (D_k + D_{m,\nu}) + (1-\alpha_{m,\nu})(1-\beta_{k,\nu}) \times (D_b + D_{m,u})]. \quad (10)$$

Finally, the weighted average of the video delivery delay on all $|\mathbf{M}|$ MEC servers is the overall average video delivery delay of the system D defined as

$$D = \frac{1}{\lambda} \sum_{m=1}^{|\mathbf{M}|} \lambda_m \times D_m \quad (11)$$

where $\lambda = \sum_{m=1}^{|\mathbf{M}|} \lambda_m$.

Moreover, we calculate the average local cache-hit ratio ϕ_l and the average cluster-hit ratio ϕ_k as follows:

$$\phi_l = \frac{1}{|\mathbf{M}|} \sum_{u\in\mathbf{U}} \sum_{\nu\in\mathbf{V}} \rho_{v,u} \times \sum_{m\in\mathbf{M}} \alpha_{m,\nu} \quad (12)$$

$$\phi_k = \frac{1}{K} \sum_{u\in\mathbf{U}} \sum_{\nu\in\mathbf{V}} \rho_{v,u} \times \sum_{k=1}^{K} \beta_{k,\nu}. \quad (13)$$

If $\alpha_{m,\nu} = \alpha \; \forall m \in \mathbf{M}, \forall \nu \in \mathbf{V} \Rightarrow \phi_l = \alpha, \phi_k \approx \alpha \times |\mathbf{M}|$.

B. Tradeoff Analysis

According to (2), the probability $\beta_{k,\nu}$ of finding the video v in the MEC servers' cluster k, is a function of the cluster size $|\mathbf{M}_k|$. In other words, if the cluster enlarges, the cached-content diversity increases and, consequently, $\beta_{k,\nu}$ increases. Hence, the average cluster-hit ratio increases with the cluster size and, consequently, the average video delivery delay decreases.

In contrast, as shown in (9), the service rate of the cluster reduces with an increase in the cluster size $|\mathbf{M}_k|$. Therefore, if the cluster enlarges, μ_k decreases and, according to (6), the intracluster delay D_k increases. As a consequence, the video delivery delay increases. This indicates a tradeoff between $\beta_{k,v}$ and D_k.

There is also a relation between the cluster size $|\mathbf{M}_k|$ and the backhaul delay D_b. When the backhaul delay is high due to a heavy traffic, enlarging the cluster results in more video requests being responded from the MEC servers at the edge and a lower number of the video requests is directed to the origin media server. Therefore, increasing the size of the cluster not only reduces the backhaul load, but also reduces the video delivery delay significantly. Nevertheless, we should determine the cluster size in such a way that the intracluster delay does not become higher than the backhaul delay.

IV. Problem Formulation

In this section, we formulate the optimization problem of the video delivery delay minimization. Regarding the tradeoffs discussed in the previous section, the video delivery delay can be expressed as a function of both the video request arrival rate and the cluster size. Moreover, the local cache-hit ratio and the cluster-hit ratio, which significantly influence the delay, are influenced by two factors: 1) the cache size of the MEC servers; and 2) the cache placement. As a result, the video delivery delay is intricately dependent on both the cache size of the MEC servers and the cache placement. Therefore, the optimization problem targeted in our article is defined as follows:

$$\underset{|\mathbf{M}_k|,\Delta_{m,\nu}}{\text{minimize}} \; D(\lambda_m, \Delta_{m,\nu}, C_m, |\mathbf{M}_k|)$$

subject to

$$1 \le K \le |\mathbf{M}| \quad (C1)$$

$$\sum_{k=1}^{K} |\mathbf{M}_k| \le |\mathbf{M}| \quad (C2)$$

$$\lambda_m \le \mu_m \quad \forall m \in \mathbf{M} \quad (C3)$$

$$\lambda_k \le \mu_k \quad 1 \le k \le K \quad (C4)$$

$$\sum_{v=1}^{|\mathbf{V}|} \Delta_{m,\nu} \times S_v \le C_m \quad \forall m \in \mathbf{M}. \quad (C5)$$

The goal of the optimization problem is to determine the cluster size($|\mathbf{M}_k|$) and the cache placement variable ($\Delta_{m,\nu}, m \in \mathbf{M}, v \in \mathbf{V}$) to minimize the video delivery delay, with the given arrival rate of video requests ($\lambda_m, \quad m \in \mathbf{M}$), and storage capacity constraints ($C_m, \quad m \in \mathbf{M}$).

The constraints $(C1)$ and $(C2)$ indicate that all active $|\mathbf{M}|$ MEC servers of the mobile networks are clustered into K disjoint groups. $(C3)$ limits that the arrival rate of the video requests in MEC servers should be lower than their video transmission rate to prevent the waiting times become infinite. Similarly, $(C4)$ limits that the aggregate arrival rate of the video requests to the clusters should be lower than their video delivery service rate for preventing the system to be overwhelmed. Note that the aggregated arrival rate is the function of the cluster size and the video requests arrival rate to the MEC servers of the cluster. $(C5)$ indicates the limitation of the storage capacity of each MEC server. As described in Section II, $\Delta_{m,\nu}, m \in \mathbf{M}, v \in \mathbf{V}$ is a binary variable which shows how the video content is cached in MEC servers of the system. Choosing video content from the video library $\mathbf{V}$ to be placed in the caches of $\mathbf{M}$ MEC servers to minimize the video delivery delay while meeting the storage capacity constraints corresponds to the classic binary knapsack problem. Therefore, this is an NP-hard problem. Moreover, the cache placement impacts on the cache-hit ratio and, consequently, the cluster-hit ratio and the service rate of the cluster, μ_k.

To make the NP-hard cache placement problem more manageable, we apply a uniform cache placement strategy that stabilizes the variable $\Delta_{m,\nu}$. Consequently, the optimization problem narrows down to finding an optimal cluster size, aiming to reduce the video delivery delay.

V. Proposed Dynamic MEC Server Clustering (DyMECC)

In this section, we describe our proposed DyMECC. First, in Section V-A, we elaborate on the effect of the load of the video requests on the service rate of the clusters that motivates the idea of DyMECC in presence of the variable arrival rate of video requests. Then, we determine the cluster size minimizing the video delivery delay in Section V-B. Last, Section V-C summarized the proposed DyMECC algorithm reducing the video delivery delay via dynamic clustering of the MEC servers.

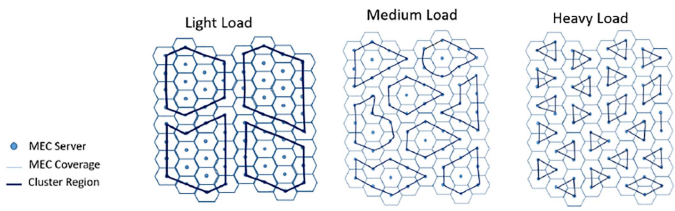

Fig. 4. MEC server clustering in our proposed DyMECC under different loads.

A. Motivation and General Idea

Since the arrival rate of the video requests to the cluster is the aggregation of the arrival rate of the video requests to the MEC servers of the cluster, the intracluster delay is multiplicatively increased by an increase in the arrival rate of the video requests to the MEC servers, especially when the clusters are large. High arrival rates result in many concurrent content deliveries, which require simultaneous usage of the Xn interfaces for content delivery path toward the serving MEC server. This creates a high demand for preempting the Xn interfaces and results in many videos being placed in the waiting queue of the Xn interfaces, causing congestion and a significant increase in delivery delay. Hence, there is a necessity for dynamic adjustment of the cluster size with respect to the arrival rate of the video requests.

When the video request rate is low (light load), the cluster can be large enough to take full advantage of the caches of the MEC severs reside in the radio access network while keeping the intracluster delay lower than the backhaul delay. On the contrary, when the request rate is high (heavy load), the cluster size should be shrunk to prevent overloading the Xn interfaces. These two basic presumptions represent fundamental aspects of the proposed DyMECC. Furthermore, DyMECC works as a congestion control mechanism to prevent overloading the Xn interfaces.

Fig. 4 illustrates an example of MEC server clustering for different arrival rates (light load, medium load, and heavy load). Thick blue lines determine the cluster region in which MEC servers belonging to the same cluster share their cached video contents to each other.

It is worth mentioning that since the MEC server clustering is done as a logical grouping for cooperative caching, and the interconnection of the MEC servers does not change, the available capacity of Xn interfaces exploited for content transferring is not affected by enlarging the cluster size. In other words, our solution does not put any additional burden on the Xn interfaces in terms of conventional exchanging of control information and just exploits the available capacity of Xn interfaces for transferring the video contents.

B. Determining the Cluster Size

In each time interval, the cluster size is derived based on the observed request arrival rate. The proposed dynamic MEC server clustering keeps the video delivery delay minimized in presence of a variable arrival rate of video request. We derive the cluster size that minimizes the video delivery delay while meeting the constraints described in Section IV.

In light load conditions, there is almost no waiting time in the delivery path of the video content from two MEC servers within the cluster. In this case, the cluster size should be determined in a way that the intracluster delay becomes lower than the backhaul delay, i.e., $D_k < D_b$. In other words, the increase in the intracluster delay is tolerable until it does not exceed the backhaul delay, because if the intracluster delay becomes higher than the backhaul delay, the MEC server clustering is not effective anymore and results in higher video delivery delay. Hence, to be ensured that the intracluster delay is always lower than the backhaul delay, we should fulfill the following condition:

$$\left(2\sqrt{|\mathbf{M}_k|_L} - 2\right) \times \frac{\frac{1}{|\mathbf{V}|}\sum_{v=1}^{|\mathbf{V}|} S_v}{R_{Xn}} < D_b \tag{14}$$

where $|\mathbf{M}_k|_L$ is the cluster size in light load conditions. If we substitute $\frac{1}{|V|}\sum_{\nu\in V} S_\nu$ with its equivalent S (the average size of the videos in the video library), we have

$$\left(2\sqrt{|\mathbf{M}_k|_L} - 2\right) \times \frac{S}{R_{Xn}} < D_b. \tag{15}$$

Moving all the variables except $|\mathbf{M}_k|_L$ to the right side of the inequality, we have

$$2\sqrt{|\mathbf{M}_k|_L} < \frac{R_{Xn} \times D_b}{S} + 2. \tag{16}$$

Finally, we get the following relation for the cluster size in light load condition:

$$|\mathbf{M}_k|_L < \frac{1}{4} \times \left(\frac{R_{Xn} \times D_b}{S} + 2\right)^2. \tag{17}$$

In addition, according to the constraint (C4) in the optimization problem, the average request arrival rate of the cluster should be lower than the average service rate of the cluster, i.e., $\lambda_k < \mu_k, \quad 1 \le k \le K$. If we substitute λ_k and μ_k with their equivalents according to the (8) and (9), respectively, and substitute the ξ_k with the average distance in the cluster k, in terms of the number of MEC servers in the shortest delivery path between any two arbitrary nodes in the cluster [32], we obtain

$$|\mathbf{M}_k|_H \times \lambda \times \left(1 - (1-\alpha)^{|\mathbf{M}_k|_H}\right) < \frac{1}{\frac{S}{R_{Xn}} \times \left(\frac{\sqrt{|\mathbf{M}_k|_H}}{2} + \frac{2}{3}\right)} \tag{18}$$

where $|\mathbf{M}_k|_H$ is the cluster size in heavy load conditions.

Since the probability of the local cache-hit is $0 \le \alpha \le 1$, we can approximate $(1-\alpha)^{|\mathbf{M}_k|_H}$ with $1 - (|\mathbf{M}_k|_H \times \alpha)$. If we neglect $\frac{2}{3}$, which is a small constant value, we get

$$\Rightarrow |\mathbf{M}_k|_H^2 \times \lambda \times \alpha < \frac{1}{\frac{S}{R_{Xn}} \times \frac{\sqrt{|\mathbf{M}_k|_H}}{2}}. \tag{19}$$

If we move all the variables of the cluster size $|\mathbf{M}_k|_H$ to the left-hand side of the inequality, we have

$$\Rightarrow |\mathbf{M}_k|_H^2 \times \sqrt{|\mathbf{M}_k|_H} < \frac{2 \times R_{Xn}}{\lambda \times S \times \alpha} \tag{20}$$

$$\Rightarrow |\mathbf{M}_k|_H < \sqrt[5]{\left(2 \times \frac{R_{Xn}}{\lambda \times S} \times \frac{1}{\alpha}\right)^2} \tag{21}$$

According to the above expression, the cluster size is the function of the ratio of the bit rate of Xn interface to the aggregate bit rate of video streams, $\frac{R_{Xn}}{\lambda \times S}$, and the inverse of the local cache-hit ratio α. The larger ratio of $\frac{R_{Xn}}{\lambda \times S}$ yields the larger cluster size, whereas the larger value of α limits the cluster size for limiting the video exchanges between the MEC servers inside the cluster. As our contribution is taking benefit of the collaborative

Algorithm 1: DyMECC Algorithm.

Input: $\mathbf{M} = \{1, 2, \ldots, |\mathbf{M}|\}$; $\mathbf{V} = \{1, 2, \ldots, |\mathbf{V}|\}$; $\mathbf{U} = \{1, 2, \ldots, |\mathbf{U}|\}$, Δt.

1: $t = 0$.
2: Initialize caches by uniform distribution of video content over MEC servers.
3: **while** $t \leq$ simulation time **do**
4: $t = t + \Delta t$.
5: Derive the cluster size in light load condition; $|\mathbf{M}_k|_L$ (17).
6: Derive the cluster size in heavy load condition; $|\mathbf{M}_k|_H$ (21).
7: Determine the cluster size at the time t; $|\mathbf{M}_k|_t = \min\{|\mathbf{M}_k|_L, |\mathbf{M}_k|_H\}$ (22).
8: Group $|\mathbf{M}|$ MEC servers into the clusters with a size of $|\mathbf{M}_k|_t$ using K-means algorithm.
9: **end while**

caching of not only neighbor MEC servers, but also other MEC servers, which are accessible via the communication interfaces until the constraints (17) and (21) are met, we determine the cluster size as follows:

$$|\mathbf{M}_k| = \min\{|\mathbf{M}_k|_L, |\mathbf{M}_k|_H\} = \min\left[\frac{1}{4} \times \left(\frac{R_{Xn} \times D_b}{S} + 2\right)^2, \sqrt[5]{\left(2 \times \frac{R_{Xn}}{\lambda \times S} \times \frac{1}{\alpha}\right)^2}\right]. \quad (22)$$

In fact, there is no specific transition point from the light load to the heavy load. More precisely, while $|\mathbf{M}_k|_L$ is lower than $|\mathbf{M}_k|_H$, the load is considered as the light load, and the cluster size is determined by $|\mathbf{M}_k|_L$. Similarly, as the load increases and $|\mathbf{M}_k|_H$ becomes lower than $|\mathbf{M}_k|_L$, the load is considered as the heavy load and the cluster size is determined by $|\mathbf{M}_k|_H$. In other words, we introduce a general formula for determining the cluster size and the load intensity is implicitly taken into account in determining the cluster size in each time interval using (22).

C. DyMECC Algorithm

In this section, we introduce the proposed algorithm, which dynamically adjusts the cluster size based on the actual arrival rate of video requests, see Algorithm 1. We distribute the video contents of the video library uniformly over the MEC servers to initialize the caches of the MEC servers (line 2). The DyMECC algorithm runs periodically every Δt (line 4) and determines the cluster size in each time interval (line 8). Then, the MEC servers are regrouped into k clusters with a size $|\mathbf{M}_k|$. The values of two limits for the cluster size are calculated based on (17) and (21) (see lines 5 and 6). The minimum of these two limits is used as the cluster size in each time interval (line 7). The derived cluster size $|\mathbf{M}_k|$ is applied to cluster the MEC servers into k groups using K-means algorithm (line 8).

VI. Performance Evaluation

In this section, we evaluate the performance of our proposed method in terms of different performance metrics, such as video delivery delay, cluster-hit ratios, and the utilization of Xn interfaces. We conduct comprehensive simulations to assess the performance of DyMECC. In Section VI-A, we provide the information about how we set up the simulation and set the simulation parameters. We briefly introduce the algorithms that we compare our proposed DyMECC against them in Section VI-B. We categorize the competitive algorithms into the two main categories. Finally, in Section VI-C, we analyze the simulation results.

TABLE II
Simulation Parameters

Simulation Parameter	Value
Number of MEC servers	144 [23]
Ratio of MEC server cache size to the video library size	0.1 [24]
Xn interface capacity	2 Gbps [29]
Average size of video contents	50 Mb [33]
Number of videos	200 [25]
Average transmission delay between MEC server and the user	1 ms [23]
Backhaul delay	300 ms [33]
Distribution of video requests	Zipf [22]
Zipf's parameter	0.8 [23]
Cache placement	Uniform [22]
Distribution of inter-arrival time of video requests	Exponential [23]
Clustering method	K-means [23]

A. Simulation Settings

We set the main system parameters based on the related works in our simulations, see Table II. Based on these parameters, the transmission delay between two neighboring MEC server, D_{Xn}, is 25 ms. The backhaul delay is considered 300 ms [33]. The intracluster delay, D_k, depends on the load and the size of the cluster k, nevertheless for the special case of the cluster size 10 and the load of 5 requests per second, it is about 70 ms. Additionally, as articulated in Section IV, we have posited a scenario where the distribution of video content across the MEC servers is uniform [22].

B. Competitive Algorithms

We compare the performance of the proposed DyMECC with existing state-of-the-art works introduced in Section I. We categorize the existing works into the following categories.

1) *Fixed clustering:* i.e., the clustering of the MEC servers is fixed during the network setup and is not adjustable with the variation of the arrival rate of video requests [22], [23], [34], [35].
2) *Noncooperative caching:* in which no cooperation among MEC servers is provided [12], [36].

In fixed clustering approaches, the cluster size is set to 10 as it is assumed in [34].

C. Simulation Results and Analysis

In this section, we provide the results of comparing our proposed algorithm with the algorithms categorized in Section VI-B. We also analyze the performance of our proposed algorithm with respect to different system parameters, i.e., cache size, video size, and Zipf parameter.

As illustrated in Fig. 5, the proposed DyMECC outperforms the competitive algorithms in terms of the video delivery delay. In a light load, the video delivery delay is reduced to about

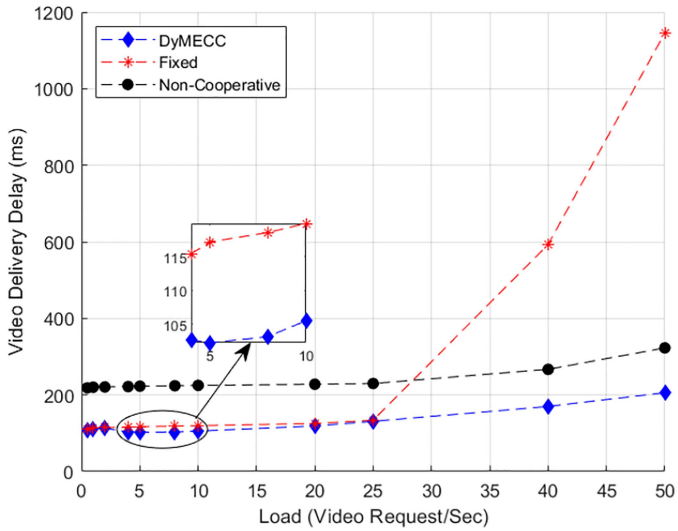

Fig. 5. Video delivery delay versus load.

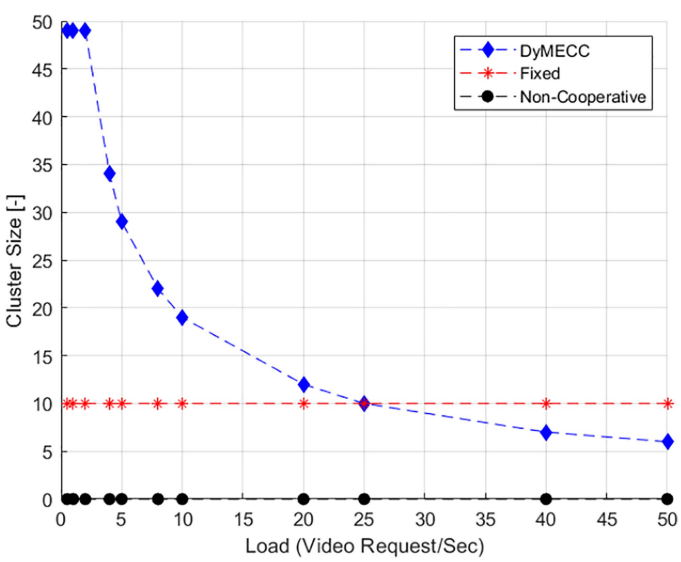

Fig. 6. Cluster size versus load.

half of the video delivery delay achieved by the noncooperative algorithms, and by 15% compared to the fixed clustering algorithms. In the fixed algorithms, when the load is high, the Xn interfaces within the cluster become saturated, and the waiting times for transferring the videos over Xn interface among MEC servers exponentially increase. Therefore, the video delivery delay drastically rises. On the contrary, since the clusters are shrinking in DyMECC when the load increases, the video delivery delay is kept at its minimum possible value. Also as evidenced, a slight increase in video delivery delay is observed in noncooperative caching due to the queuing delay on the serving MEC server, particularly in heavy load conditions.

As illustrated in Fig. 6, the DyMECC dynamically adjusts the cluster size according to the variable load of video requests. When the load is low, the cluster size is determined based on (17). In the light load, DyMECC determines the cluster to be big enough to maximize the cluster-hit ratio while intracluster delay does not exceed the backhaul delay. In such case, DyMECC minimizes the total number of clusters needed to handle the

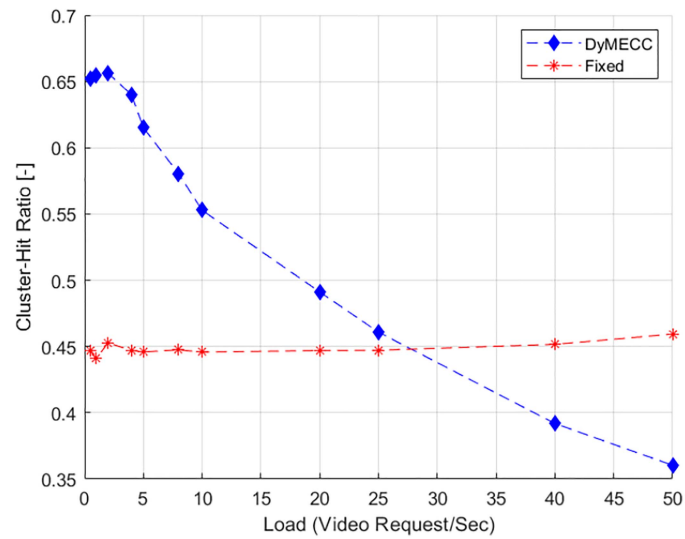

Fig. 7. Cluster-hit ratio versus load.

users' requests. As the load increases, (21) determines the cluster size because the minimum value of the cluster size derived from (17) and (21) is used to determine the cluster size, as shown in (22). As the load continues to increase, the cluster size may become even lower than the cluster size used in the fixed algorithms, especially when the load exceeds 25 video requests per second. This demonstrates the superiority of the proposed DyMECC algorithm over the fixed algorithms in terms of adapting to variable loads and reducing the cluster size to handle any load.

The cluster-hit ratio remains nearly constant when the cluster size does not change over time as shown in Fig. 7, while the cluster-hit ratio in DyMECC changes with respect to the cluster size which is dynamically determined at each time interval. When the load is light, the clusters are big, and most of the video requests are responded within the cluster that leads to a lower video delivery delay compared to the fixed methods. As the load increases, the cluster size shrinks. As a result, in heavy load conditions, the clusters become small and consequently the cluster-hits decrease. Although decreasing the cluster-hit ratio seems unpleasant, it helps to reduce the content delivery delay in heavy load conditions. As we discussed the tradeoffs in Section III-B, when the load is heavy, fewer cluster-hits result in shorter waiting time, and lower the intracluster delay, which is the most significant contributor to the video delivery delay. Therefore, by reducing the intracluster delay, the overall video delivery delay can be reduced. The utilization of Xn interfaces is illustrated in Fig. 8. If the cluster size is fixed, the utilization of Xn interfaces increases almost linearly in relation to the load. When the utilization of Xn interfaces exceeds 0.5, the waiting time increases drastically and, consequently, the video delivery delay increases as well. Although our proposed DyMECC provides low video delivery delay under different load intensities, its efficiency is more considerable in heavy loads. In fact, DyMECC acts as a congestion avoidance mechanism to prevent over-utilizing the capacity of Xn interfaces. Since the cluster size is dynamically adjusted according to the load in DyMECC, when the load is low, the cluster becomes large enough to efficiently take advantage of the available capacity of Xn interfaces to

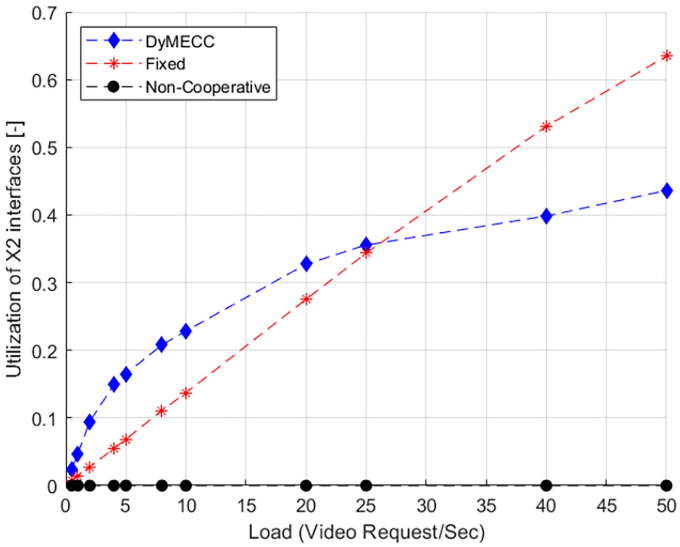

Fig. 8. Utilization of Xn interfaces versus load.

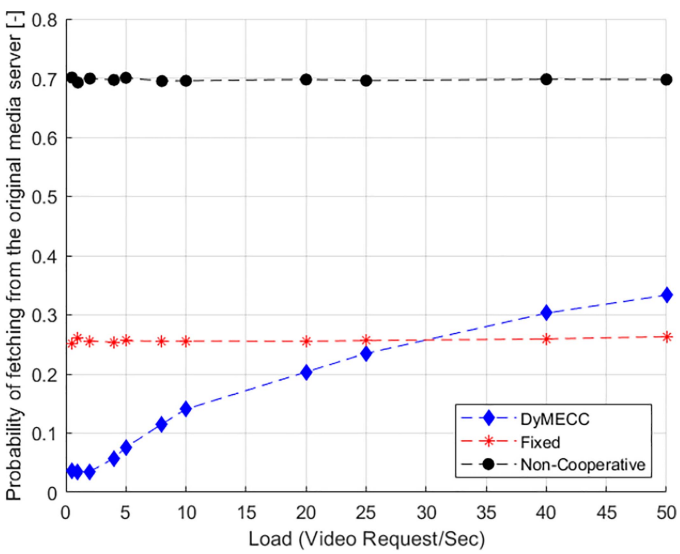

Fig. 9. Probability of fetching the requested video from the original media server versus load.

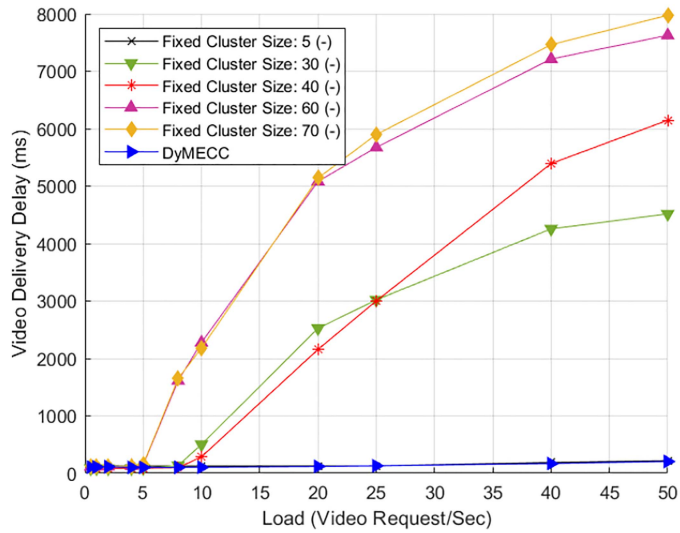

Fig. 10. DyMECC versus different fixed cluster sizes.

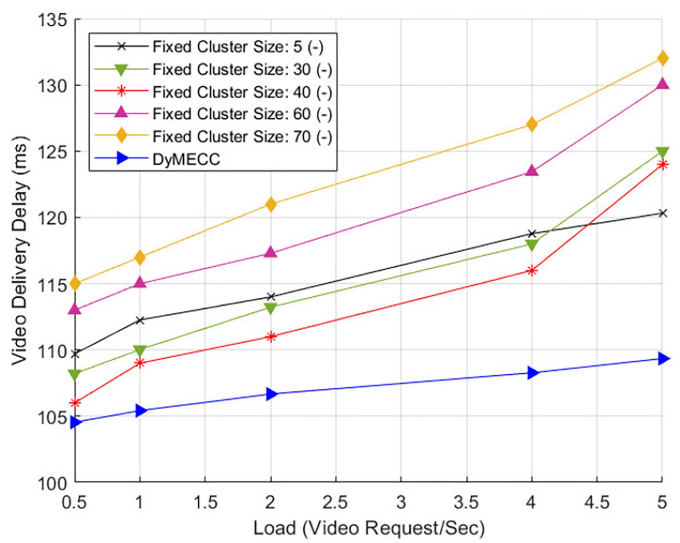

Fig. 11. DyMECC versus different fixed cluster sizes under light load condition.

reduce the video delivery delay. Thus, in light load, the utilization of Xn interfaces in DyMECC is higher than in fixed algorithms. As the load increases, the utilization of Xn interfaces in DyMECC decreases in comparison to the fixed algorithms due to the mechanism of the congestion avoidance in DyMECC.

Fig. 9 shows the probability of fetching the requested video from the original media server. In DyMECC, the requested video is only rarely (below 20% of the time) received from the original media server via the backhaul when the load is not heavy. When the load becomes heavy and the clusters shrink, more videos are received from the original media server. However, the video delivery delay is still much lower than in the fixed algorithms.

To more effectively demonstrate the performance improvement of our proposed DyMECC over existing methods, we conduct a comparative analysis with various Fixed clustering schemes that employ different fixed cluster sizes. As illustrated in Fig. 10, the saturation point of the system corresponds to the cluster size, which highlights the limit of the cluster service rate with respect to the aggregated arrival rate of the cluster. In contrast, the congestion avoidance feature of DyMECC ensures that the video delivery delay does not lead to a drastic increase, and hence ensures smooth video streaming. The fixed clustering schemes, which rely on predetermined cluster sizes, often face limitations in terms of adaptability to varying load conditions. As a result, the fixed clustering is not efficient in case of the network congestion. This is shown in Fig. 10, where the saturation occurs earlier for systems with larger fixed cluster sizes, leading to a rapid deterioration in video delivery quality.

To facilitate the comparison of the performance of DyMECC with various fixed clustering schemes under light load conditions, we zoom a part of Fig. 10 for loads lower than 5 requests per second in Fig. 11. As depicted in Fig. 11, when the cluster sizes are greater or smaller than the optimal cluster size determined by the DyMECC algorithm (for example the optimal cluster size of 49 for arrival rate of 2 video requests per second, see Fig. 7), the video delivery delay is higher than that observed when using the proposed DyMECC. This

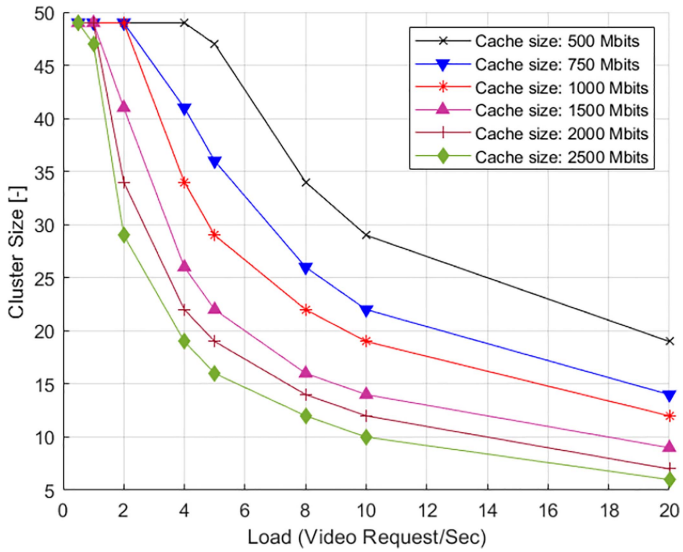

Fig. 12. Cluster size versus load under different cache sizes.

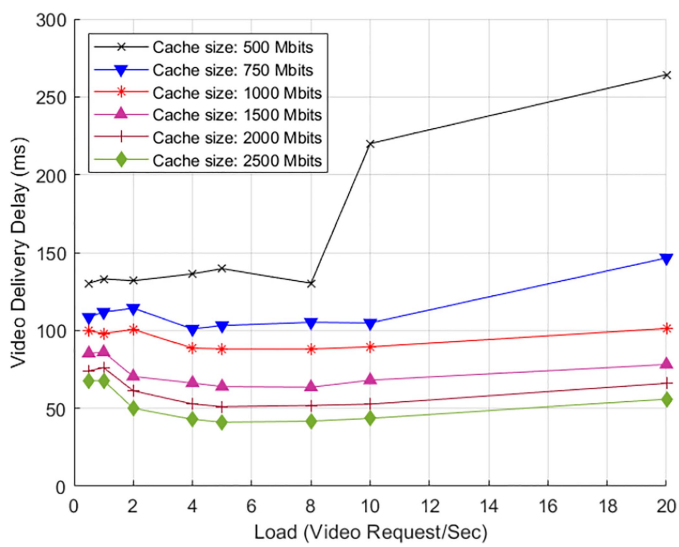

Fig. 13. Video delivery delay versus load under different cache sizes.

highlights the relation between the intracluster delay and the cluster hit-ratio, emphasizing the importance of selecting an appropriate cluster size for efficient video delivery. Moreover, Fig. 11 shows that the video delivery delay for the cluster sizes of 30 and 40 exceeds that of the cluster size of 5 when the load reaches 5 requests per second. This observation further indicates that larger cluster sizes tend to cause the system to saturate more rapidly, leading to a deterioration in video delivery performance. These results emphasize the benefits of the DyMECC algorithm in determining an optimal cluster size that balances the intracluster delay and cluster-hit ratio.

As discussed in Section V, the probability of finding the video content in the MEC server, α, is one of the main factors in determining the cluster size in each interval. Having a fixed number of videos in the video library, α is a function of cache size. In Fig. 12, we demonstrate that the cluster size is dynamically determined in our proposed method with respect

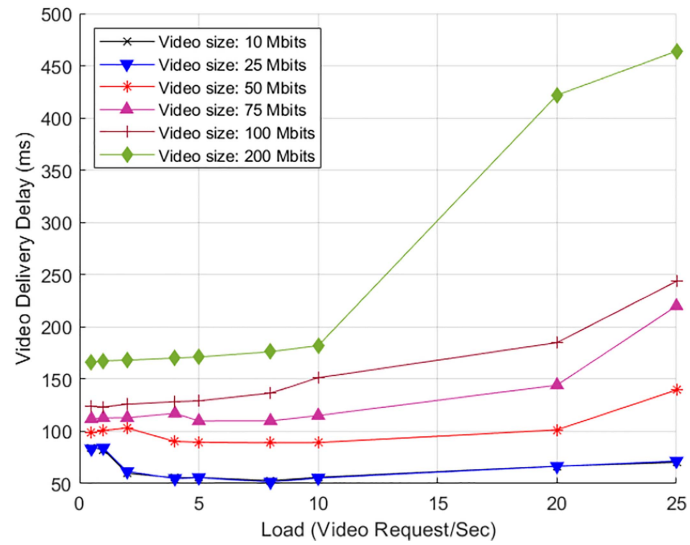

Fig. 14. Video delivery delay versus load under different video sizes.

to the cache size, under different loads. Smaller cache sizes yield bigger clusters, because the cluster-hits smoothly increases with the cluster size for the small cache sizes. Moreover, since concurrent video deliveries within the cluster are limited by the Xn interface capacity, when the load is high and the cache size is large, the cluster size is shrunk. For example, when the load is 20 requests per second and the cache size is 2500 Mbits, each cluster consists of seven MEC servers, whereas for the cache size of 500 Mbits, the cluster size becomes nine.

Furthermore, we investigate the effect of key parameters (cache size, Zipf parameter, Xn interface capacity, and video size) of the system on the video delivery delay in our proposed DyMECC. In Fig. 13, we show that DyMECC keeps the video delivery delay low in different configurations of cache sizes even if the load increases. However, the minimum delay is different in each case. The reason behind the sharp increase in the video delivery delay for the cache size 500 Mbits and the loads greater than 10 requests per second is that the demand for using the communication interfaces within the cluster is beyond the capacity of Xn interfaces.

In addition, the video size is one of the key parameters that affects the video delivery delay. As shown in Fig. 14, larger videos result in longer video delivery delays. For example, when the video size is 200 Mbits, the Xn interfaces within the cluster become quickly over-utilized if the load increases. As discussed in Section III-B, to balance between the cluster-hit ratio and the intracluster delay, DyMECC switches from $|\mathbf{M}_k|_H$ to $|\mathbf{M}_k|_L$ for determining the cluster size (see Algorithm 1). This results in a reduction of the video delivery delay. In fact, we sacrifice the cluster-hit ratio to reduce the intracluster delay by dwindling the cluster size aimed to minimize the video delivery delay (See Figs. 6 and 7 at the same time).

We also analyze the effect of the Zipf parameter of the distribution of the video requests on the video delivery delay in Fig. 15. A higher Zipf parameter results in a lower video delivery delay. This is because, for the higher values of the Zipf parameter, especially for the values greater than one, the distribution of video requests becomes closer to the uniform

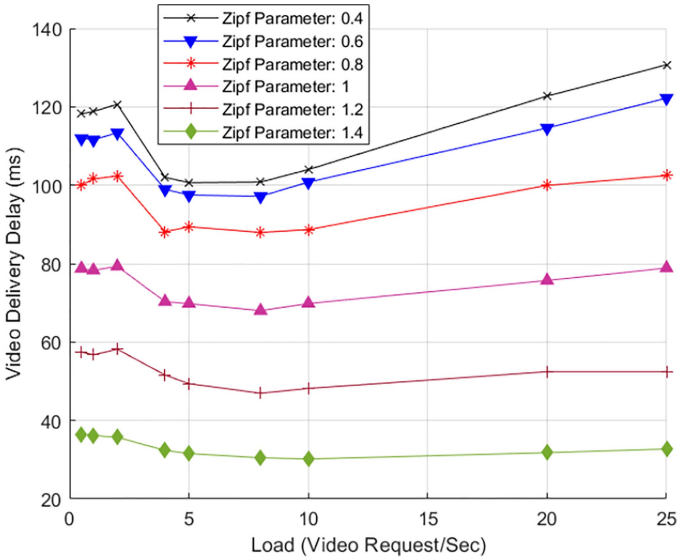

Fig. 15. Video delivery delay versus load under different Zipf parameters.

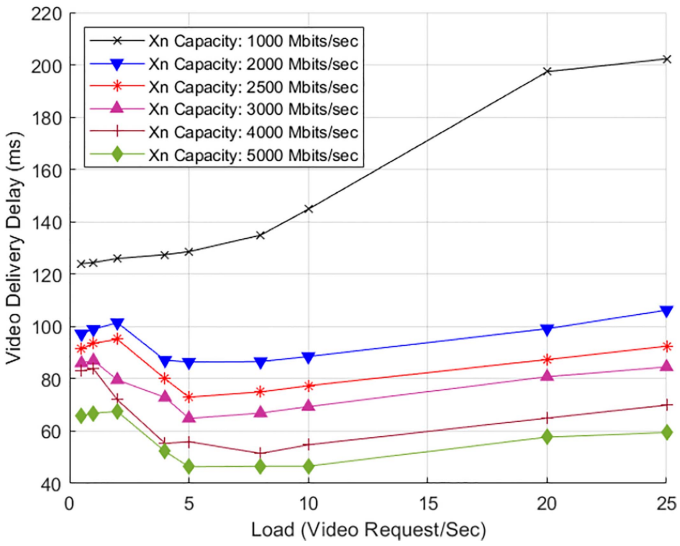

Fig. 16. Video delivery delay versus load under different Xn capacities

distribution. Additionally, a nearly sharp reduction in the video delivery delay is observable while the load increases. The reason is that during our proposed method of dynamic clustering, there is a point at which the first reduction in cluster size occurs (e.g., from the cluster size of 49 to the cluster size of 34 for the load of 4 requests per second in our configuration, see Fig. 6) to prevent the Xn interfaces to be overutilized.

Moreover, we delve deeper into the impact of Xn interface capacity on the video delivery delay. As depicted in Fig. 16, the performance of our proposed DyMECC algorithm improves significantly as the capacity of the Xn interface increases. This is an encouraging observation, as the development of high-bandwidth Xn interfaces can further enhance the efficiency of MEC server clustering for video content distribution. Additionally, DyMECC can effectively accommodate the evergrowing demand for higher video quality. For a capacity of 1 Gbps, the Xn interface reaches close to its maximum utilization more quickly compared to other values of Xn capacity. This can be attributed to the fact that, with higher capacities, the queuing time for transferring video content within the cluster is significantly reduced. As a result, the communication interfaces among the MEC servers can handle more data without becoming congested, leading to lower video delivery delays. This finding highlights the importance of incorporating high-capacity Xn interfaces in mobile networks, as they enable more efficient MEC server clustering and improve video content delivery performance. Furthermore, this suggests that future mobile networks should continue to invest in the development and deployment of advanced communication interfaces to accommodate the increasing demand for high-quality video content and ensure seamless video delivery to the end users.

VII. Conclusion

We have proposed a dynamic MEC server clustering scheme to minimize the video delivery delay. DyMECC adopts the cluster size based on the actual workload of video requests. The simulation results prove that DyMECC outperforms the existing fixed algorithms in terms of video delivery delay, especially when the arrival rate of video requests is high. DyMECC reduces the video delivery delay by more than 15% over existing fixed algorithms while prevents Xn interfaces to be saturated.

In future, we plan to extend our proposed DyMECC to take into account the computational capacity of MEC servers, which can be exploited for online video transcoding to dynamically determine the cluster size. Moreover, we intend to adapt DyMECC for scenarios that unmanned aerial vehicle-based cache-enabled MEC servers provide video delivery services to the users.

References

[1] Ericsson Public Information, "Ericsson mobility report Q4 2022 update," Technical Report, 2022. [Online]. Available: https://www.ericsson.com/491da6/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-q4-2022.pdf

[2] Cisco and I. Cisco Systems, "Cisco annual internet report (2018–2023)," Cisco White Paper, San Jose, CA, USA, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] C. and I. C. Systems, "Cisco visual networking index: Forecast and trends, 2017–2022 White Paper," *Cisco Forecast Methodol.*, vol. 17, pp. 2017–2022, 2019.

[4] Z. Zhang et al., "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, Sep. 2019.

[5] P. Yang, Y. Xiao, M. Xiao, and S. Li, "6G wireless communications: Vision and potential techniques," *IEEE Netw.*, vol. 33, no. 4, pp. 70–75, Jul./Aug. 2019.

[6] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5G networks: A. deep learning based approach," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Serv.*, 2018, pp. 1–6.

[7] Y. Liu, S. Wang, M. S. Obaidat, X. Li, and P. Vijayakumar, "Service chain caching and workload scheduling in mobile edge computing," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4389–4400, Sep. 2022.

[8] F. Brunero and P. Elia, "Fundamental limits of combinatorial multi-access caching," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 1037–1056, Feb. 2023.

[9] Z. Sang, S. Guo, Q. Wang, and Y. Wang, "GCS: Collaborative video cache management strategy in multi-access edge computing," *Ad Hoc Netw.*, vol. 117, 2021, Art. no. 102516.

[10] C. Fang et al., "Cache-assisted content delivery in wireless networks: A new game theoretic model," *IEEE Syst. J.*, vol. 15, no. 2, pp. 2653–2664, Jun. 2021.

[11] C. Li, Y. Zhang, M. Song, X. Yan, and Y. Luo, "An optimized content caching strategy for video stream in edge-cloud environment," *J. Netw. Comput. Appl.*, vol. 191, 2021, Art. no. 103158.

[12] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Netw.*, vol. 32, no. 6, pp. 50–57, Nov./Dec. 2018.

[13] S. Rezvani, S. Parsaeefard, N. Mokari, M. R. Javan, and H. Yanikomeroglu, "Cooperative multi-bitrate video caching and transcoding in multicarrier NOMA-assisted heterogeneous virtualized MEC networks," *IEEE Access*, vol. 7, pp. 93511–93536, 2019.

[14] D. Tsigkari and T. Spyropoulos, "Caching and recommendation decisions at transcoding-enabled base stations," in *Proc. IEEE Glob. Commun. Conf.*, Rio de Janeiro, Brazil, 2022, pp. 147–153.

[15] C. Li, H. Zhao, Y. Zhao, B. Zhang, and C. Li, "Joint transcoding- and recommending-based video caching at network edges," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4928–4937, Sep. 2022.

[16] M. Yeh, C.-H. Wang, D.-N. Yang, and W. Liao, "Efficient caching for 360° videos with dynamic view selection," in *Proc. IEEE/CIC Int. Conf. Commun. China*, 2019, pp. 225–230.

[17] M. Yeh, C.-H. Wang, D.-N. Yang, J.-T. Lee, and W. Liao, "Mobile proxy caching for multi-view 3D videos with adaptive view selection," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2909–2921, Aug. 2022.

[18] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 80–87, Jun. 2018.

[19] P. Lin, K. S. Khan, Q. Song, and A. Jamalipour, "Caching in heterogeneous ultradense 5G networks: A comprehensive cooperation approach," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2. pp. 22–32, Jun. 2019.

[20] K. Zhang, J. Cao, S. Maharjan, and Y. Zhang, "Digital twin empowered content caching in social-aware vehicular edge networks," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 239–251, Feb. 2022.

[21] Z. Gu, H. Lu, and C. Zou, "Horizontal and vertical collaboration for VR delivery in MEC-enabled small-cell networks," *IEEE Commun. Lett.*, vol. 26, no. 3, pp. 627–631, Mar. 2022.

[22] K. Bilal, E. Baccour, A. Erbad, A. Mohamed, and M. Guizani, "Collaborative joint caching and transcoding in mobile edge networks," *J. Netw. Comput. Appl.*, vol. 136, pp. 86–99, 2019.

[23] D. Ren, X. Gui, K. Zhang, and J. Wu, "Hybrid collaborative caching in mobile edge networks: An analytical approach," *Comput. Netw.*, vol. 158, pp. 1–16, 2019.

[24] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug. 2018.

[25] Z. Chen, J. Lee, T. Q. S. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3401–3415, May 2017.

[26] R. Chen, H. Lu, and P. Ma, "User-centric cooperative MEC service offloading," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2021, pp. 1–6.

[27] Y. -H. Chiang, W. Liao, and Y. Ji, "RELISH: Green multicell clustering in heterogeneous networks with shareable caching," in *Proc. IEEE Glob. Commun. Conf.*, 2018, pp. 1–7.

[28] S. Song, C. Lee, H. Cho, G. Lim, and J. M. Chung, "Clustered virtualized network functions resource allocation based on context-aware grouping in 5G edge networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1072–1083, May 2020.

[29] G. S. Park and H. Song, "Cooperative base station caching and X2 link traffic offloading system for video streaming over SDN-enabled 5G networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 2005–2019, Sep. 2018.

[30] S. Zhou et al., "Caching in dynamic environments: A. near-optimal online learning approach," *IEEE Trans. Multimedia*, vol. 25, pp. 792–804, 2023.

[31] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 3098–3130, Fourthquarter 2018.

[32] R. Luo, D. Belis, R. M. E. Amiee, and G. A. Manson, "Estimation of average hop count using the grid pattern in multi-hop wireless ad-hoc network," in *Proc. London Commun. Symp.*, 2002, pp. 1–4.

[33] M. F. Tuysuz and M. E. Aydin, "QoE-based mobility-aware collaborative video streaming on the edge of 5G," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7115–7125, Nov. 2020.

[34] Y. Zhou, Z. Zhao, R. Li, H. Zhang, and Y. Louet, "Cooperation-based probabilistic caching strategy in clustered cellular networks," *IEEE Commun. Lett.*, vol. 21, no. 9, pp. 2029–2032, Sep. 2017.

[35] Y. Sun, Z. Chen, and H. Liu, "Delay analysis and optimization in cache-enabled multi-cell cooperative networks," in *Proc. IEEE Glob. Commun. Conf.*, 2016, pp. 1–7.

[36] Y. Dong, S. Guo, Q. Wang, S. Yu, and Y. Yang, "Content caching-enhanced computation offloading in mobile edge service networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 872–886, Jan. 2022.

Ali Doostmohammadi received the B.Sc. degree in computer engineering from the Ferdowsi University of Mashhad. Mashhad, Iran, and the M.Sc. degree in computer engineering from the Iran University of Science and Technology, Tehran, Iran, in 2004 and 2008, respectively. He is currently working toward the Ph.D. degree in reducing video delivery delay in 6G mobile networks using mobile edge computing with the the Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.

He is also with 6Gmobile Research Laboratory, Czech Technical University, Prague, Czech Republic, since 2021. His research interests include computer networks, mobile edge computing, and collaborative caching in mobile networks.

Mohammad Reza Khayyambashi was born in Isfahan, Iran in 1961. He received the B.Sc. degree in computer hardware engineering from Tehran University, Tehran, Iran, and the M.Sc. degree in computer architecture from the Sharif University of Technology (SUT), Tehran, Iran, and the Ph.D. degree in computer engineering, distributed systems from University of Newcastle upon Tyne, Newcastle upon Tyne, England, U.K., in 1987, 1990, and 2006, respectively.

He is currently working as an Associate Professor with the Department of Computer Architecture, Faculty of Computing Engineering, University of Isfahan, Isfahan, Iran. He has successfully supervised graduated Ph.D. and M.Sc. students and is currently supervising Ph.D. students in his research area. His research interests include distributed systems, computer networking, software defined network, mobile computing, Internet of Things (IoT), and edge/fog/cloud computing.

Naser Movahedinia received the B.Sc. degree in electrical engineering from Tehran University, Tehran, Iran in 1987, the M.Sc. degree in electrical and communication engineering from Isfahan University of Technology, Isfahan, Iran in 1990, and the Ph.D. degree in system engineering, telecommunication from Carleton University, Ottawa, Canada in 1997.

He is currently a Full Professor with the Faculty of Computer Engineering, University of Isfahan. His research interests include wireless networks, artificial intelligence in communication, and Internet Technology.

Zdenek Becvar (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in telecommunication engineering from Czech Technical University in Prague, Czech Republic, in 2005 and 2010, respectively.

He has authored or coauthored more than 100 conference and journal papers and he is the (co-)inventor of four US patents. He is currently an Associate Professor with the Department of Telecommunication Engineering, Czech Technical University in Prague, and he leads 6Gmobile Research Lab at the same university. From 2006 to 2007, he was with Sitronics R&D Center, Prague, focusing on speech quality in VoIP. Furthermore, he was involved in research activities of Vodafone R&D Center, Czech Technical University in Prague, in 2009. He was on internships with Budapest Politechnic, Budapest, Hungary, in 2007, CEA-Leti, Grenoble, France, in 2013, and EURECOM, Biot, France, in 2016 and 2019. From 2013 to 2017, he was a representative of the Czech Technical University in Prague in ETSI and 3GPP standardization organizations.