Journal of Network and Computer Applications 168 (2020) 102737

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

RAN-aware adaptive video caching in multi-access edge computing networks

Shashwat Kumar[a,b,*], Sai Vineeth Doddala[c], A. Antony Franklin[a], Jiong Jin[b]

[a] *Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, India*
[b] *School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, VIC 3122, Australia*
[c] *Samsung R&D Institute, Bangalore, India*

ARTICLE INFO

Keywords:
Multi-access Edge Computing (MEC)
Radio Network Information (RNI)
Video caching

ABSTRACT

Videos are expected to be a primary contributor to an anticipated massive surge in mobile network data. Caching the videos within the mobile network can significantly reduce the network load and Operational Expenditure (OPEX) for mobile network operators. Multi-access Edge Computing (MEC) can enable the video caching by providing processing and storage capabilities within the network. However, content providers usually employ Dynamic Adaptive Streaming over HTTP (DASH) for video streaming, which contains multiple bit-rate representations of videos. Constrained by its capacity, MEC can not cache all representations of popular videos. Video transcoding mitigates this issue to a certain extent by converting the higher available video bit-rate to a requested lower one; but, it can quickly exhaust the available edge processing power by transcoding a large number of videos in parallel. Therefore, caching appropriate video bit-rates that can serve the maximum number of users in the network is a non-trivial problem. To resolve this problem and to efficiently utilize the resources (processing and storage) at the network edge, we took a non-traditional approach for video caching that utilizes the network information provided by MEC's Radio Network Information (RNI) Application Program Interface (API). In particular, RNI API provides Radio Access Network (RAN) status information that can be employed to estimate the probability distribution of requested video qualities. In this work, we formulate the video caching problem as an Integer Linear Programming (ILP) for the hit-rate maximization. Since the optimization problem requires the knowledge of all future requests, it obviously cannot be used in real-time. Therefore, we develop a RAN-aware Adaptive VidEo cachiNg (RAVEN) method that uses network information to make an informed decision for video bit-rate selection in video caching coupled with transcoding and maximizes the number of served users form the network edge. Simulation results demonstrate that the RAVEN significantly outperforms state-of-the-art algorithms in the domain and performs closer to the optimal solution.

1. Introduction

According to the Cisco Visual Networking Index (VNI) forecast (Forecast, 2019), smartphone data traffic will surpass the PC traffic by 2022 and account for 44% of the total IP traffic. By 2020, mobile data will reach 77.5 exabytes per month and the existing cellular network backhaul will face severe congestion if Mobile Network Operators (MNOs) do not take some preventive measures. Interestingly by 2022, IP video traffic will account for 82% of all the IP traffic. MNOs should optimize video content delivery in particular to avoid backhaul congestion. One simple method to alleviate network congestion, due to data surge, is network capacity expansion, which is not always viable and also requires a significant capital investment. Obviously, network operators need alternative methods to deliver this vast amount of data without congestion. In-network-caching is a promising solution to handle backhaul congestion, which caches the popular content within the cellular network. Other than avoiding congestion, caching the frequently accessed content within the network also reduces the transit service payments to Internet Service Providers (ISP). It thus reduces the operational cost of the cellular network and improves the user's quality of experience through fast content delivery by avoiding long-distance transmission (Belshe, 2010). Various architectures such

* Corresponding author. Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, India.
E-mail addresses: cs15resch11011@iith.ac.in (S. Kumar), cs14btech11011@iith.ac.in (S.V. Doddala), antony.franklin@iith.ac.in (A.A. Franklin), jiongjin@swin.edu.au (J. Jin).

https://doi.org/10.1016/j.jnca.2020.102737
Received 5 November 2019; Received in revised form 15 April 2020; Accepted 31 May 2020
Available online 4 July 2020

as Content-Centric Networking (Jacobson et al., 2009), Information-Centric Networking (Abdullahi et al., 2015), and Multi-access Edge Computing (MEC) (ETSI - Introductory Technical White Paper) enable the MNOs to cache the content within the network. Being one of the key technologies for future 5G networks (ETSI - MEC in 5G Networks) and its compatibility with contemporary cellular networks, MEC befits in-network-caching in the cellular network. The European Telecommunications Standards Institute (ETSI) standardized MEC to be compatible with the Network Function Virtualization (NFV) framework, which makes it incredibly versatile for the deployment of services or applications in cellular networks. MEC brings the cloud computing capabilities (i.e., storage and computation) at the network edge (e.g., base station), which enables the support for Ultra-Reliable Low Latency Communication (URLLC) and high bandwidth applications. Moreover, through various Application Program Interfaces (API) such as Radio Network Information (RNI), MEC opens the possibility for intelligent services that can make an informed decision using network and context information. Conceiving it as an integral part of future cellular networks, we adapted the MEC architecture in this work.

Most of the video streaming services use adaptive video streaming to serve a diverse set of users with a good quality of experience. Adaptive video streaming methods such as Dynamic Adaptive Streaming over HTTP (DASH) store different bit-rate versions of a video on the server. For a video request, based on user device characteristics and available bandwidth, an appropriate video bit-rate is selected to ensure the best user experience. Video content providers (e.g., YouTube) offer multiple (five to six) bit-rate representations of a video to befit the user requirements. If an MEC server at the edge of the network caches all the bit-rate versions of the videos, the storage space of the MEC will not be sufficient. For this reason, even with the known popularity distribution, the system cannot cache all representations of the popular videos and thus video caching poses a unique challenge compared to caching the content which has only one representation. Video transcoding is able to solve this problem to some extent by converting a higher bit-rate version of a video to a desired lower bit-rate version (Kumar and Vineeth, 2018). Hence, by caching only the highest bit-rate version and using the transcoding to serve the lower bit-rate video requests, the storage exhaustion problem could be mitigated. However, transcoding is a computation intensive task and transcoding multiple video streams in parallel can itself exhaust the limited computational resources at the MEC. Consequently, it is important to find an effective solution that utilizes the computation and storage resources on MEC by caching the appropriate bit-rates of the videos. To solve the processing power exhaustion problem without caching all bit-rate version of videos, a few selected bit-rate versions of a video should be cached to avoid transcoding for each lower bit-rate video request. In this case, selecting an appropriate video bit-rate version for caching is a nontrivial problem, because bit-rate selection, in adaptive video streaming, dominantly depends upon the available network throughput to the client. If the available throughput is known to the cellular network, this information can be utilized to make a wise video caching decision. RNI API (ETSI - Radio Network Information API) of MEC (ETSI - Technical Requirements) makes network information available to various services deployed on the MEC platform. Applications or services deployed on MEC are able to measure the available throughput of a user in the cell by using RNI API, and same can be utilized to estimate the expected video bit-rate of the user requests. In this work, we formulate the caching problem as an Integer Linear Program (ILP) that maximizes the cache hit ratio subject to the cache capacity and processing power constraints of the MEC. It is not possible to solve the ILP for each user request or the requests in a time slot in real time, because it takes a significant time due to its NP-completeness. Hence, we design a RAN-aware Adaptive VidEo cachiNg (RAVEN) method that exploits the RNI API to determine the probabilistic video bit-rate to make caching decision. The RAVEN method attains a balance in the processing power and storage utilization at MEC by selecting appropriate bit-rate versions of videos to cache. The main contributions of this paper are summarized as follows:

- We introduce a central cache catalog, namely, central cache manager, which ensures the collaboration and cache consolidation among the MEC servers. It helps in avoiding the data duplication in MEC cache network, and enables the storage and computation resources sharing among MEC servers.
- We measure the scalability of the standalone transcoding solution on a testbed to investigate its feasibility.
- We formulate the multibit-rate video caching and transcoding as an ILP based optimization problem with an objective of maximizing the hit ratio in the MEC cache network.
- We design a RAN-aware adaptive video caching (RAVEN) method that takes advantage of the RNI API to estimate the requested video bit-rate and utilizes this information for cache replacement decisions. The RAVEN method reduces the backhaul traffic and consequently reduces the OPEX (operational cost) for the mobile network operators.

The rest of the paper is organized as follows. Previous works on content caching are discussed in Section 2. In Section 3, we explain the motivation and define the problem statement. Section 4 highlights the details of the proposed RAVEN method. Results based on extensive simulations are presented and analyzed in Section 5. Finally, we conclude the work in Section 6.

2. Related work

Over the years, many solutions have been set forth to solve the network congestion problem. Content Delivery Networks (CDNs) have been presented as a viable solution to minimize the network congestion by storing the data near the consumers (Liu et al., 2016). Other solution like Information Centric Networks (ICNs) (Kutscher et al., 2016; Abdullahi et al., 2015; de Assunção et al., 2018) and Data-Oriented Network Architecture (DONA) (Koponen et al., 2007) tried to address the network congestion by redesigning the network itself. Some authors suggested the deployment of CDN in the cellular network to serve content to the users from within the mobile network (Liu et al., 2016). By doing so, MNOs can reduce the load on backhaul links and the operational cost of downloading data through ISPs. But being the primary cause for network congestion, it is essential to optimize the video traffic, and most of the solutions do not consider the unique attributes of video data. Therefore, they are not very effective in handling the congestion.

Edge technologies such as MEC and fog computing offer solutions to deploy micro cloud services at the edge of the network (ETSI - Introductory Technical White Paper; Hu et al., 2017). With the increasing demand for video streaming, various solutions have been introduced to cache the data in the cellular network (Shanmugam et al., 2013; Pedersen et al., 2016; Tran et al., 2017a). MEC offers the storage and computation capabilities at the edge of the network (ETSI - Introductory Technical White Paper). But, MEC servers have limited computation and storage resources; therefore, these resources should be engaged with diligence. Considering the limited resources on MEC, collaborative caching has been proposed to increase the video hit-ratio and share the resources among the MEC servers (Tran et al., 2017b). Gharaibeh et al. (2016) came up with a collaborative caching algorithm to minimize the total cost paid by the content providers in a multi-cell coordinated system. Ostovari et al. considered unlimited cache space in their model to minimize the aggregated caching and download cost for corroborative caching (Ostovari et al., 2016). In Bastug et al. (2014), files are proactively cached during off-peak periods based on popularity, correlations among users, and file access patterns and these files are disseminated to user's social ties via D2D communications. For edge caching, Wang et al. (2014) deployed the content-centric networking within the cellular network to minimize the delay and traffic load. Shanmugam et al., 2013 used the femtocell-like base stations for caching with weak backhaul links but large storage capacity.

Zhang et al. (2018) proposed a cooperative edge caching architecture with the help of mobile edge computing but they did not consider the different representation (multiple bit-rates) of same data. Bilal et al., 2019 proposed a collaborative caching and transcoding scheme to minimize the CDN cost and video access delay. Irrespective of network conditions, their algorithm, based on processing availability, fetches the higher video bit-rates and transcode it to serve the users and uses the LRU approach for cache replacement. In contrast, the proposed method utilizes the RAN information to improve caching performance and employ a profit-based strategy for cache replacement. Moreover, our method delivers substantial improvement without the use of transcoding, as well. Mehrabi et al. (2018) formulate the QoE-traffic optimization with collaborative edge caching as an integer non-linear programming (INLP) optimization problem. They use network information for bit-rate adaptation and proposed a retention based caching algorithm. In contrast to their work, we use the network information for video bit-rate selection in cache replacement and propose an efficient use of transcoding to serve different bit-rates to the users. Tran and Pompili (2019) proposed proactive cache placement and request scheduling scheme. Their caching algorithm only works when video popularity is known when popularity is not known they use the LRU algorithm. On the other hand, our proposed method is a reactive caching scheme and works well with known and unknown popularity. One other difference is the fact that we use the cache consolidation along with collaborative caching to boost the performance.

Most of the works mentioned above deal with the caching in which content has only one version to be cached, however, the Adaptive Bit Rate (ABR) protocol such as DASH supports multiple bit-rate versions of a video for different network conditions. Therefore, video caching becomes more challenging and requires intelligent decision making to select appropriate bit-rate versions of the video for caching. Transcoding a video from higher to lower bit-rate version is one of the solutions to handle the multiple versions of the same video content. Shen et al. (2004) discussed various techniques to transcode a video from higher bit-rate to lower bit-rate version and deduced that compressed-domain based approaches such as bit-rate reduction and spatial resolution reduction are most effective. Pederson et al. proposed caching and processing for multi-bitrate video streaming (Pedersen et al., 2016), unfortunately they do not consider the collaborative scheme of multiple caching servers. In Joint Collaborative Caching and Processing (JCCP) (Tran et al., 2017b), a collaborative caching and transcoding solution is introduced in which MEC servers collaborate to share the cached content and perform the transcoding if a higher bit-rate version is available in the cache. However, their scheme considers replication of the content among the cache servers that results in storage waste; furthermore, JCCP does not utilize the bit-rate information in caching decision as well. On contrary, we design a consolidated caching method in which content is not replicated within the MEC cache network and it capitalizes on the network information to select the appropriate bit-rate version for caching.

3. Motivation, system architecture, and problem formulation

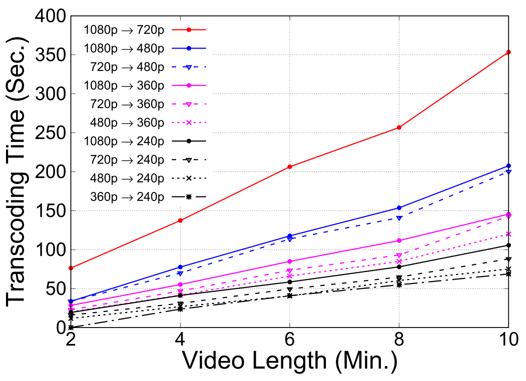
3.1. Motivation

A multifold increase in the data passing through the mobile network is anticipated, and video data is expected to be a primary contributor to it (Forecast, 2019). Caching the videos within the mobile network, with the help of MEC, can help in reducing the network load and OPEX for mobile network operators. However, in video streaming, content can have multiple representations (bit-rates) and traditional caching schemes are agnostic to it. MEC, at the network edge, can not cache all representations of popular videos because of capacity constraints. Therefore, caching appropriate bit-rates of the videos, which can serve the maximum number of users, is a vital decision that impacts the caching performance. Video transcoding, which can convert a high bit-rate video to lower bit-rate, provides an ingenious solution to this issue. However, exclusively depending on the transcoding, by only caching the highest available bit-rate video, may lead to a processing power crunch on MEC servers.

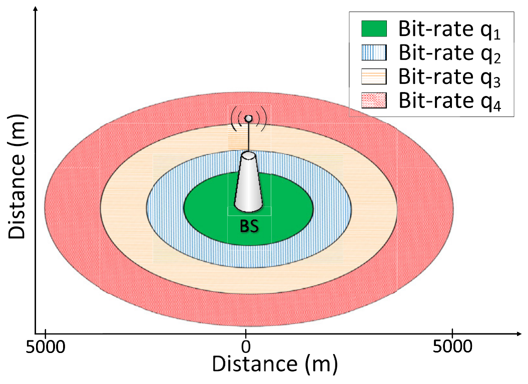
We empirically assess video transcoding scalability on a simple testbed using *ffmpeg* (FFmpeg) without using any specialized hardware for video encoding. The main objective is to evaluate the video transcoding scalability, and not to observe the correlation between engaged CPU cores and number of parallel transcodings, therefor the evaluation is carried out only with two CPU configurations. In Table 1,

Table 1
Possible number of real-time parallel transcodings from higher to lower resolution.

	Using 8 Cores (2.4 Ghz)				Using 4 Cores (2.4 Ghz)			
	720p	480p	360p	240p	720p	480p	360p	240p
1080p	3	6	10	14	2	3	4	6
720p	–	7	12	18	–	4	6	9
480p	–	–	14	22	–	–	8	11



(a)



(b)

Fig. 1. (a) Time taken to transcode videos of different playtime from a higher bit-rate input video to a lower bit-rate video. (b) Supportable video bit-rate ($q_1 > q_2 > q_3 > q_4$) based on the UE distance from the cellular Base Station (BS).

the first column of the table contains the source video bit-rate and the second row indicates the target bit-rate for transcoding, and the values in the cells present the number of feasible parallel transcodings. There are two noteworthy deductions from the results in Table 1: first, the number of parallel transcodings depends on the source and target video bit-rates; second, video transcoding is not very scalable because in the best-case scenario ($480p \rightarrow 240p$ transcoding) the system can serve only 22 users.

Furthermore, Fig. 1(a) unveils that it takes less time to transcode from a closest higher bit-rate version e.g., transcoding to 240p target bit-rate from higher bit-rate sources (1080p, 720p, 480p, and 360p), $360p \rightarrow 240p$ transcoding takes the minimum time. Therefore an adequate video bit-rates selection, for caching, should maximize the number of served users with optimal use of transcoding. In a mobile network, UEs' downlink throughput, which is utilized to determine the video bit-rate, profoundly depends on its distance from the base station. We simulate an LTE network in NS3 and results in Fig. 1(b) shows that the user who is further away from the BS tends to have lesser throughput and, consequently, a lower video bit rate. Network information utilization can improve video caching performance. For example, if most of the users are at the cell edge, then they are likely to get a lower downlink throughput and expected to consume the video in a lower bit-rate version. So, in this scenario, it is not desirable to cache a higher bit-rate video at the network edge, which might be requested only by some users who are near the base station and getting a higher bandwidth. Since resources at the edge are limited, bit-rate agnostic caching solutions are not effective for video caching. Using RNI API, MEC can provide the network information, which can significantly enhance the cache performance. In this work, we propose a solution that uses the network information to cache appropriate video bit-rates coupled with transcoding to improve the video caching performance.

3.2. System architecture

Fig. 2 depicts the system architecture, where MEC servers are deployed alongside the eNBs in a cellular RAN, providing computation and storage capabilities to enable caching at the edge of the network. MEC servers collaborate to share their computing and storage resources. If an MEC server caches a video, then the other MEC servers are able to fetch the video and serve the users without caching it locally. The cache architecture has the following components:

Cache Storage: Cache storage caches the videos for future user requests and managed by the local cache manager.

Local Cache Manager: Local cache manager resides on the MEC server at the network edge. On receiving a video request, local cache manager searches for the video in the local cache on the MEC server. If the video is not available locally, the local cache manager sends the request to the central cache manager for the video location (MEC server) and forwards the received location to the streaming server. When a video is cached or gets replaced in the local cache, the local cache manager propagates this information to the central cache manager.

Central Cache Manager: Central cache manager keeps track of all the videos that are cached in the MEC network. Central cache manager empowers the collaboration and cache consolidation among the MEC servers to avoid replication of the same video in the cache network.

Streaming Server: The streaming server on each MEC server streams the requested videos to the users by fetching the videos from the cache. If the streaming server fetches the video from the main content server, then the streaming server caches the video in the local cache.

Transcoder: Transcoder converts a video segment from high bit-rate to a requested lower bit-rate version. After transcoding, transcoder forwards the video segments to the streaming server to serve the user.

3.3. Problem formulation

Definitions of the key notation used in this work are given in Table 2. $\mathcal{N}$ denotes the set of all eNBs with MEC deployment. Cache capacity of an MEC server j is M_j, where $j \in \mathcal{N}$ and C_j denotes utilized cache storage. A user sends request for video $v_q : v \in \mathcal{V}$ in quality $q : q \in \mathcal{Q}$ at server j and the size of a video v of quality q is denoted with $r_{v_q} : r_{v_q} > 0$ (as size of a file is always positive number). The probability of a user, in an eNB, requesting the video of quality $q : q \in \mathcal{Q}$ is p_q that can be computed with the help of RNI API. The probability of a user requesting a video $v : v \in \mathcal{V}$, p_v, is measured by video popularity distribution. Either content provider can provide the video popularity distribution or MEC be able to learn it based on user requests.

The variable $c_j^{v_q} \in \{0, 1\} : j \in \mathcal{N}, v \in \mathcal{V}$, and $q \in \mathcal{Q}$ indicates if a video v in quality q is cached on an MEC server j. If MEC server j caches the video v_q then $c_j^{v_q} = 1$, otherwise $c_j^{v_q} = 0$. To show how (with or without transcoding) and from where the video is fetched to serve a user request v_q, we define two binary variables ρ and ζ. If MEC server j serves the video request v_q from its cache or cache of MEC server k then $\rho_j^{v_q} = 1$ or $\rho_k^{v_q} = 1 : j, k \in \mathcal{N}, j \neq k$ respectively. Binary variable $\zeta_x^{yv_{q'}} = 1$ when a video $v_{q'}$ is fetched from MEC x and transcoded at server y

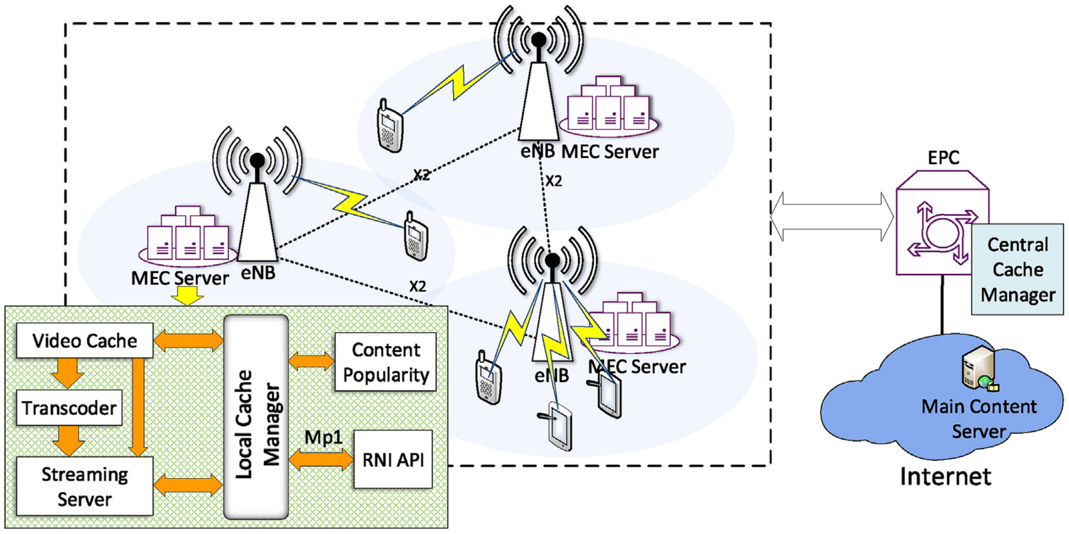


Fig. 2. MEC video caching system architecture.

Table 2
Notation.

Representation	Meaning
$\mathcal{V}$	Set of all video files
$\mathcal{Q}$	Set of available qualities
$\mathcal{N}$	Set of cache nodes
$\mathcal{R}_j$	Set of request on eNB j
$Q_{\max}, Q_{\min}$	Highest and lowest available video bit-rate
N_j	Cache node located at eNB j
P_j	Processing power capacity at MEC on eNB j
$\underline{P_j}$	Unused/available processing power at MEC on eNB j
$P_{q'\to q}$	Processing power required to transcode video from bit-rate q' to q where $q < q'$
$P_{\min}$	Minimum processing required to perform transcoding
p_q	Probability that a user will request for video in quality q
p_v	Probability of a user requesting the video v
r_{v_q}	Size of the video v in quality q
M_j	Caching capacity of MEC server j
C_j	Filled cache size of MEC on eNB j ($C_j \le M_j$)
$c_j^{v_q}$	Indicate that video v in quality q is cached at MEC server j
$\rho_x^{v_q}$	Indicate that video v_q is fetched from MEC server x
$\zeta_x^{yv_{q'}}$	Indicate that video $v_{q'}$ is fetched from MEC server x and transcoded at MEC server y to serve user request v_q

to serve a the video request v_q otherwise $\zeta_x^{yv_{q'}} = 0$. When serving MEC j performs the transcoding, to serve video request v_q, after fetching the video $v_{q'}$ from its cache or some other server k then respectively $\zeta_j^{jv_{q'}} = 1$ or $\zeta_k^{jv_{q'}} = 1 : q' > q$. In other case, when MEC server k transfers the requested video v_q to serving MEC j after locally transcoding from $v_{q'}$ then $\zeta_k^{kv_{q'}} = 1 : q' > q$. If video request v_q can not be served locally then it is fetched from the main content server which is represented by $O^{v_q} = 1$. When a user requests a video, system fulfills the request following only one of the above cases, and the following constraint enforce this condition

$$\rho_j^{v_q} + \zeta_j^{jv_{q'}} + \sum_{k\neq j}(\rho_k^{v_q} + \zeta_k^{kv_{q'}} + \zeta_k^{jv_{q'}}) + O^{v_q} = 1; \forall j,k \in \mathcal{N}, v \in \mathcal{V}$$

The system objective is to serve most of the requests from the MEC servers hence to maximize the hit ratio. A video request that is served by MEC servers in the network is regarded as a cache hit, and if the content server serves the content, a cache miss. The objective function of the formulation for a cache hit maximization is as follows

$$\text{Cache Hits (H)} = \sum_{j\in\mathcal{N}}\sum_{v_q\in R_j}(\rho_j^{v_q} + \zeta_j^{jv_{q'}} + \sum_{k\neq j}(\rho_k^{v_q} + \zeta_k^{kv_{q'}} + \zeta_k^{jv_{q'}}))$$

We model the video caching as a hit-rate maximization problem, so the MEC server at the network edge can serve most of the user requests. Available processing power and cache size on MEC servers are imposed as constraints for the formulation. The objective function in Eq. (1a) represents the number of cache hits. Constraints in Eqs. (1b) and (1c) make sure that decision variables ($\rho_j^{v_q}$ and $\rho_k^{v_q}$) are set to one only if v_q is cached on MEC server j or k, respectively. The Constraints in Eqs. (1d), (1e) and (1f) ensure that $\zeta_j^{jv_{q'}}$, $\zeta_k^{kv_{q'}}$, and $\zeta_k^{jv_{q'}}$ set to one, when video v is cached in higher bit-rate version q' and need to be transcoded to the requested bit-rate q. The Constraint in Eq. (1g) administrator

$$\text{maximize} \quad \sum_{j\in\mathcal{N}}\sum_{v_q\in R_j}(\rho_j^{v_q} + \zeta_j^{jv_{q'}} + \sum_{k\neq j}(\rho_k^{v_q} + \zeta_k^{kv_{q'}} + \zeta_k^{jv_{q'}})) \tag{1a}$$

subject to

$$\rho_j^{v_q} \le c_j^{v_q}, \qquad \forall j \in \mathcal{N}, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1b}$$

$$\rho_k^{v_q} \le c_k^{v_q}, \forall k \in \mathcal{N}, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1c}$$

$$\zeta_j^{jv_{q'}} \le \min(1, \sum_{q'=q+1}^{Q_{\max}} c_j^{v_{q'}}), \forall j \in \mathcal{N}, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1d}$$

$$\zeta_k^{jv_{q'}} \le \min\left(1, \sum_{q'=q+1}^{Q_{\max}} c_k^{v_{q'}}\right), \forall j,k \in \mathcal{N}, j\neq k, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1e}$$

$$\zeta_k^{kv_{q'}} \le \min\left(1, \sum_{q'=q+1}^{Q_{\max}} c_k^{v_{q'}}\right), \forall k \in \mathcal{N}, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1f}$$

$$\rho_j^{v_q} + \zeta_j^{jv_{q'}} + \sum_{k\neq j}\left(\rho_k^{v_q} + \zeta_k^{kv_{q'}} + \zeta_k^{jv_{q'}}\right) + O^{v_q} = 1, \quad \forall j,k \in \mathcal{N}, j\neq k, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1g}$$

$$\sum_{v_q\in R_j} r_{v_q} c_j^{v_q} \le M_j, \forall j \in \mathcal{N}, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1h}$$

$$\sum_{v_q\in R_j} P_{q'\to q}\left(\zeta_j^{jv_{q'}} + \sum_{k\neq j, k\in\mathcal{N}} \zeta_k^{jv_{q'}}\right) + \sum_{k\neq j, k\in\mathcal{N}}\sum_{v_q\in R_k} P_{q'\to q}\zeta_j^{jv_{q'}} \le P_j, \quad \forall j,k \in \mathcal{N}, j\neq k, v \in \mathcal{V}, q\prime, q \in \mathcal{Q}, q' > q \tag{1i}$$

$$\rho_j^{v_q}, \rho_k^{v_q}, \zeta_j^{jv_{q'}}, \zeta_k^{kv_{q'}}, \zeta_k^{jv_{q'}}, O^{v_q} \in \{0,1\}, \forall j,k \in \mathcal{N}, j\neq k, v \in \mathcal{V}, q \in \mathcal{Q} \tag{1j}$$

that content is fetched from only on place; either from an MEC server in the network or the main server. The Constraint in Eq. (1h) makes sure that the system does not violate the cache capacity, and the constraint in Eq. (1i) puts upper bound on consumed processing power. The formulated problem is an ILP and proved to be NP-Complete in Theorem 3.1, therefore it cannot be solved in polynomial time. Moreover, knowledge of all the future video requests is required to solve the ILP and that is unattainable, so formulated ILP cannot be solved in real time. Therefore, we design a method which uses the RNI and video popularity distribution and makes the caching decision in real time. The offline solution of ILP is regarded as an optimal benchmark to compare the performance of the proposed method.

Theorem 3.1. *The video caching problem in* Eq. (1a) *is NP-complete.*

Proof. Feasibility of any given solution of Eq. (1a) can be checked in polynomial time, thus the problem is NP. Now, we show that problem is NP-hard through the reduction of an instance of 0–1 knapsack problem, which is known to be NP-complete, to our problem. In the collaborative environment, a set of MEC servers on eNBs can be considered as a knapsack with the capacity equivalent to cache size, the profit of caching (users served from the edge) a video as the item's value, and storage space requirement for caching as the weight of the items. Then the knapsack problem is naturally transformed into the problem of maximizing caching benefits of MEC servers. Hence the 0–1 knapsack problem can be reduced to the problem defined by Eq. (1a), with a one-to-one mapping between a set of MEC servers and knapsack. Therefore the solution of Eq. (1a) will correspond to the solutions of the knapsack problem.

Algorithm 1 RAVEN Algorithm.

```
Input: User request ($v_q$) for a video $v$ in quality $q$ on MEC server j.
Initialize: Available processing power $\overline{P}_1, \dots, \overline{P}_N = P$.
Initialize: Cache on MEC servers $C_1, \dots, C_N = \varphi$
1:  for $j \in 1, \dots, N$ do
2:    for each request $v_q$ on MEC j do
3:      if $c_j^{v_q} == 1$ then    # $v_q$ is cached on serving MEC.
4:        Serve the user from the cache on MEC j.
5:      else if $c_k^{v_q} == 1; k \neq j$ then
6:        Fetch the video from MEC server $k$ and serve the user.
7:      else if $c_j^{v_{q'}} == 1$ and $P_{q' \to q} < \overline{P_j}; q' > q$ then
8:        Transcode the video ($v_{q'}$ to $v_q$) on MEC $j$ and serve the user.
9:        $\overline{P_j} = \overline{P_j} - P_{q' \to q}$    # for transcoding duration
10:       CacheReplacement($v_q, C_j$)
11:     else if $c_k^{v_{q'}} == 1$ and $P_{q' \to q} < \overline{P_k}; q' > q, k \neq j$ then
12:       Fetch the video from MEC $k$ after transcoding ($v_{q'}$ to $v_q$).
13:       $\overline{P_k} = \overline{P_k} - P_{q' \to q}$    # for transcoding duration
14:       CacheReplacement($v_q, C_j$)
15:     else if $c_k^{v_{q'}} == 1$ and $P_{q' \to q} < \overline{P_j}; q' > q, k \neq j$ then
16:       Fetch the video from MEC $k$ and transcode ($v_{q'}$ to $v_q$) on MEC $j$.
17:       $\overline{P_j} = \overline{P_j} - P_{q' \to q}$    # for transcoding duration
18:       CacheReplacement($v_q, C_j$)
19:     else
20:       Fetch the video ($v_q$) from content server and serve the user.
21:       CacheReplacement($v_q, C_j$)
22:     end if
23:   end for
24: end for
```

4. RAN-aware adaptive VidEo cachiNg (RAVEN)

We propose RAVEN for multi-bitrate video caching in the cellular network through MEC. The proposed caching algorithm makes the caching decision based on the value of caching profit, where profit represents the expected number of satiable requests on caching a video in a given bitrate. The video popularity, network conditions, and processing power availability affect the value of caching profit. RAVEN avoids the content replication in the caching network by employing cooperation among the MEC servers which enables a MEC to serve the videos that are cached on the other MEC servers in the network. The proposed algorithms strive to maximize the hit-ratio by caching the videos with higher profits.

Algorithms 1 and 2 represent the procedures used in RAVEN for online caching and transcoding. The algorithm starts with empty caches and video catalogs and P unit of available processing power. On a request (v_q) of video v in quality q on MEC server at eNB j, Algorithm 1 first checks if v_q is cached on MEC j (line 3). If the video is available in the local cache ($c_j^{v_q} == 1$) then it is served to the users locally. Otherwise, the algorithm searches for v_q on the other MEC servers with the help of central cache manager. If the video is cached in one of the other MEC servers ($c_k^{v_q} == 1; k \neq j$), then the video is fetched from there and served to the user (line 5). If the video is not available on any of the MEC servers in the requested quality, then the algorithm searches for a higher bit-rate (q') version of the same video (v) on the local MEC server (j) (line 7). When the video is cached ($c_j^{v_{q'}} == 1; q' > q$) locally in higher bit-rate and MEC server has sufficient available processing power ($\overline{P_j} > P_{q' \to q}$) to transcode the video, then user is served after transcoding of the video to the bit-rate q. If neither the video is cached (in requested bit-rate) on local MEC server nor the local MEC server have the processing power to transcode the video (when a higher bit-rate video is cached), then using central cache manager algorithm searches for the video on the other MEC servers in the network (line 10). If the video is available on one of the other MEC servers in higher bit-rate, then it is transcoded either on local MEC server or on the MEC server where it is cached based on processing power availability (preferably on the other MEC server) and streamed to the user (line 14). If the requested video is not available on any of the MEC servers in the network, either in the same bit-rate or higher bit-rate, then the video is fetched from the content server and cache replacement algorithm (Algorithm 2) is used to make the caching decision (line 19).

Algorithm 2 RAVEN's Cache Replacement Algorithm.
Input: New video (v_q) v in quality q and current cache (C_j).

```
1:  if $(C_j + r_{v_q}) \le M_j$ then    # If cache has empty space
2:    $c_j^{v_q} = 1$
3:    $C_j = C_j + r_{v_q}$
4:  else if $\overline{P_j} \ge P_{\min}$ then
5:    Find $q'':q'' \le q$ and $\sum_{i=q''}^{q} P_{q \to i} \le \overline{P_j}$
6:    Calculate $d_{qq''}^{v}$    # indirect profit for caching video $v$
7:    $d_{\min}^{v'} = \min C_j^d$    # cached video $v'$ with minimum profit value
8:    if $d_{\min}^{v'} < d_{qq''}^{v}$ then    # checking if profit for $v$ is greater than $v'$
9:      Replace $v'$ with $v_q$
10:   end if
11:  else if $\overline{P_j} \le P_{\min}$ then    # processing power is not available.
12:    Calculate $d_q^v$    # direct profit for caching video $v$
13:    $d_{\min}^{v'} = \min C_j^d$
14:    if $d_{\min}^{v'} < d_v^q$ then
15:      Replace $v'$ with $v_q$
16:    end if
17:  end if
```

For cache replacement, we introduce a profit based cache replacement algorithm (Algorithm 2) in which the profit is defined as hits per consumed resources (storage and processing). On a cache miss, the algorithm replaces the least profit cached video with new video if it has a higher profit value. When an uncached video is requested, then the algorithm calculates the hit-rate profit for caching that video utilizing the video popularity and probability of getting the video request in same or lower quality. If the profit of the video is higher than the least profit cached video, then the algorithm performs the replacement otherwise discard the video after serving the user. The profit for caching a video is the ratio of number of requests (either for the same or lower video bit-rate) that can be served by caching the video and the weighted sum of normalized values of required storage space and processing power (to transcode the video to lower bit-rate versions to fulfill the future requests). The algorithm considers two types of profits, namely, direct and indirect profit when replacing a video. Direct profit represents the normalized probability of a video being requested by a user in the same bit-rate, and indirect profit represents the normalized probability of video being requested in either the same or lower bit-rate version. Direct profit is defined as

$$d_q^v = \frac{p_v p_q}{\hat{r}_{v_q}} \tag{2}$$

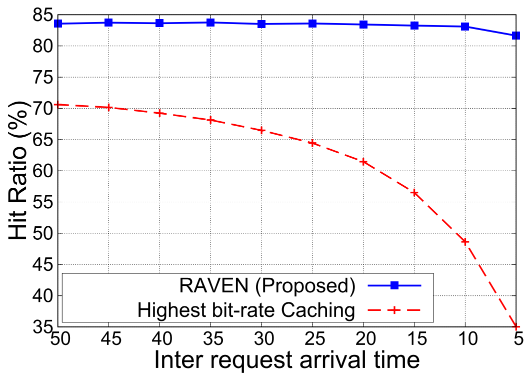
where $\hat{r}_{v_q}$ is the normalized size of the video v in quality q and p_q is the probability of a user in the cell requesting a video in the quality q. p_q is calculated with the help of RNI API on MEC (ETSI - Radio Network Information API). MEC measures the throughput of all UEs in a cell using RNI API, and the highest supported video bit-rate is assessed using these throughput values. A probability distribution, $p_q : \sum_{q=Q_{\min}}^{Q_{\max}} p_q = 1$, of requested video bit-rates is estimated using throughput data for all users in a cell, and it depends on the users distribution in the cell, e.g., if most of the users are at the edge of the cell then probability of a request begin in lower bit-rate is high. If processing power is not available for transcoding, then only direct profit is considered for cache replacement. In indirect profit, processing cost for transcoding a cached video is also considered along with the storage cost. Indirect profit is defined as

$$d_{qq''}^v = \frac{p_v \sum_{i=q''}^{q} p_i}{\alpha \hat{r}_{v_q} + \beta \sum_{i=q''}^{q} \hat{P}_{q\to i}} \tag{3}$$

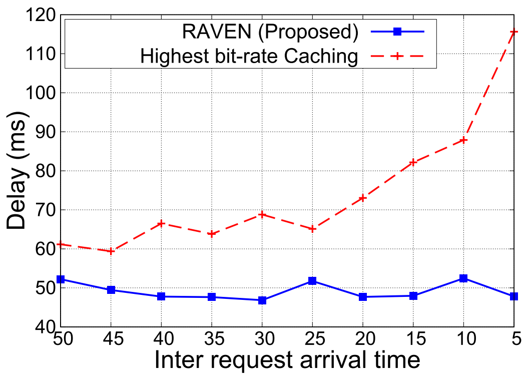
where $\hat{P}_{q\to i}$ is normalized processing power, $\hat{r}_{v_q}$ is the normalized size to cache the video v in quality q, and α, β are weight parameters.

Table 3
Simulation parameters.

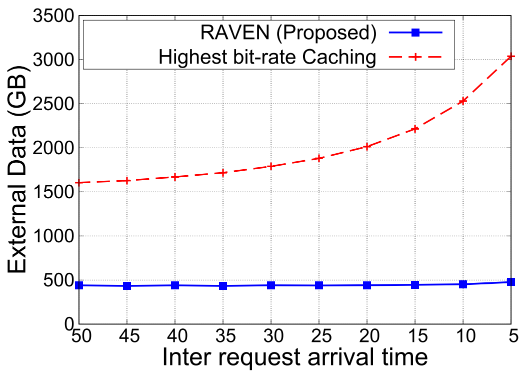
Parameter	Value
No. of MEC servers	4
UEs per MEC	20 (Uniformly distributed in the cell)
Number of videos	1000
Video bit-rates	0.4 Mbps, 1.2 Mbps, 2.5 Mbps, and 5 Mbps
Video size	50 MB, 80 MB, 100 MB, and 150 MB
Delay between MECs	[20, 60] ms
Delay between MECs and Content server	[100, 200] ms
Processing power (P)	1
Zipfs' parameter (α)	0.8
Transmission power	46 dBm
Channel Bandwidth	20 MHz
Operating Frequency (freq)	5.3 GHz
Path loss model (3GPP)	$36.7\log_{10}(d[m]) + 22.7 + 26\log_{10}(freq[GHz])$



(a) Hit Ratio



(b) Delay



(c) External Data

Fig. 3. Change in hit ratio, delay, and external data on varying the inter request arrival rate (λ).

Algorithm 2 illustrates the procedure of cache replacement algorithm. On a cache miss, for a user request of video v_q, the algorithm checks if available processing power $(\overline{P})$ is greater than minimum processing power $(P_{\min})$ for transcoding (line 6). If the MEC has enough processing power, then algorithm finds a bit-rate level (q'') up to which MEC can transcode from requested bit-rate (q) with available processing power (line 7). Afterward, it calculates the indirect profit $(d^v_{qq''})$ and finds a cached video (v'_q) with the minimum profit value $(d^{v'_q}_{\min})$. If the profit for v_q is greater than v'_q, then it replaces the later with v_q (line 10). If the MEC does not have enough processing power for transcoding (line 13), then the algorithm calculates the direct profit (d^v_q) and compares it with the minimum profit among the cached videos for replacement (line 16). Theorem 4.1 shows that the time complexity of RAVEN is $\mathcal{O}(\lceil\frac{C_j}{r_{\min}}\rceil)$. Furthermore, RAVEN attains a 2-approximation ratio for video caching, which can be established as follows. Each MEC server in the caching network makes the caching decision locally while sharing the cache information with other MEC servers. Each video, in a given bitrate, is associated with a profit value, and MEC servers successively cache the most profitable videos. Therefore, it is a cooperative caching algorithm that is local-greedy. Contrary to a central server populating all the caches, under RAVEN, each MEC server locally takes the caching decision to maximize the global hit-ratio; hence RAVEN is distributed in this sense. This sort of local-greedy algorithm ascertained to achieve a 2-approximation ratio (Borst et al., 2010).

Theorem 4.1. *Complexity of RAVEN is* $\mathcal{O}(\lceil\frac{C_j}{r_{\min}}\rceil)$.

Proof. Algorithm 1 executes for every user request. If video is cached on the MEC server in requested bit-rate, it can be searched in constant time $\mathcal{O}(1)$ using hash tables; therefore, the algorithm takes a constant time $\mathcal{C}$ for this step. If video is not cached in requested bit-rate on the MEC server then after fetching the video Algorithm 2 is executed for cache replacement. In lines 7, 9, and 15, Algorithm 2 searches for minimum profit video in the cache, which contributes to its time complexity. In worst case, line 7 carries up to $|\mathcal{Q}|$ iterations, and lines 9 and 15 require up to $\lceil\frac{C_j}{r_{\min}}\rceil$ iterations where C_j is cache size of MEC server j and $r_{\min}$ is size of a video in lowest bit-rate. Therefore the worst-case run time complexity of proposed algorithm is $\mathcal{O}(|\mathcal{Q}| + \lceil\frac{C_j}{r_{\min}}\rceil)$ where $|\mathcal{Q}|$ is the number of available video bit-rates and usually a small number (i.e., 5). Hence the complexity can be reduced to $\mathcal{O}(\lceil\frac{C_j}{r_{\min}}\rceil)$.

5. Results and discussions

In simulations, we consider an Urban Macro (UMa) cell model with four eNBs, each eNB using transmission power of 46 dBm and 20 MHz of channel bandwidth. Each eNB serves 20 active users that are uniformly distributed in a cell of 5 km radius. An MEC server is deployed at each eNB to offer the caching and processing resources. Video library $\mathcal{V}$ consists of 1000 videos, playtime of each video is 5 min, a video can be served in any of the available four bit-rate variants (0.4 Mbps, 1.2 Mbps, 2.5 Mbps, and 5 Mbps, for 360p, 480p, 720p, and 1080p video resolutions, respectively), and size of different bit-rate variants of a video are 50 MB, 80 MB, 100 MB, and 150 MB. Each user generates video requests independently following Poisson process with mean inter request interval of 8 min. The user's video requests follow the Zipf's popularity distribution (Zink et al., 2009) which determines the probability of next request to be for ith popular video. It is given by Eq.

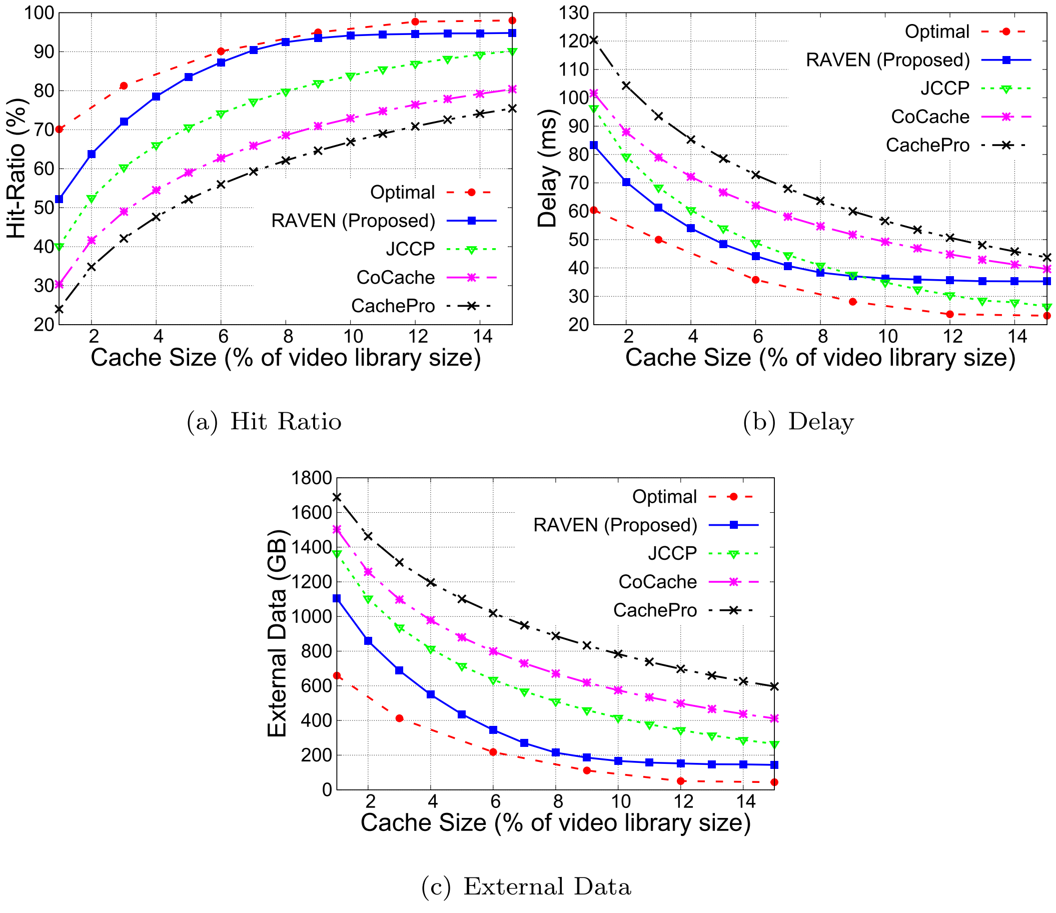


Fig. 4. Change in hit ratio, delay, and external data with increase in cache size.

(4) and used to calculate the value of p_v.

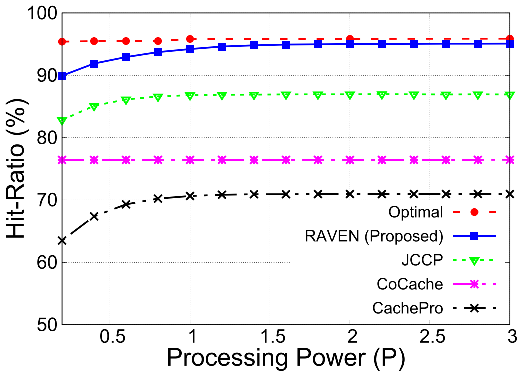
$$p_i = \frac{i^{-\alpha}}{\sum_{j=1}^{|\mathcal{V}|} j^{-\alpha}} \tag{4}$$

where $\alpha = 0.8$ is Zipf parameter (Shanmugam et al., 2013) and $|\mathcal{V}|$ is total number of videos in the library. UMa path loss model with a non-line-of-site (NLOS) condition as specified in 3GPP TR36.814 V9.2.0 (3GPP) is used to calculate the available Down-Link (DL) throughput of the users in the cell. Expected video bit-rate of user requests is estimated by utilizing the UEs' throughput. The probability distribution of video bit-rates, p_q, is measured by employing the expected bit-rate of all the users in the cell, which is used to calculate the direct and indirect profit of caching a video. On MEC platform, a service or an application can calculate the UEs' throughput using RNI API over the Mp1 reference point (ETSI - Radio Network Information API). For simplicity, we consider the stationary users and stable network conditions; users send requests for the entire videos in specific bit-rates, which is selected based on available bandwidth. By changing the granularity of users' requests to video chunks, the proposed method can be used for time-varying video bit-rates where caching decision are made for each video chunk instead of the entire video. The end-to-end latency for fetching the video content is randomly assigned between [5, 15] ms uniformly for local eNB, [20, 60] ms for neighboring MEC servers, and [100, 200] ms for origin content server or CDN on the Internet. Cache size represents the percentage of video library size that can be cached (e.g., 10% cache size means cache on each MEC server is able to store 10% of the video library). The processing power of 1 ($P = 1$) symbolizes the transcoding capability of an 8-core machine. Values from Table 1 determines how many parallel transcodings are possible, and what fraction (one 1080$p \rightarrow$ 240p transcoding takes 1/14 of the processing power of an 8-Core machine) of processing power (P) is consumed for a transcoding operation. In Table 3, we define the simulation parameters used for the evaluation. Following metrics are considered for performance evaluation:

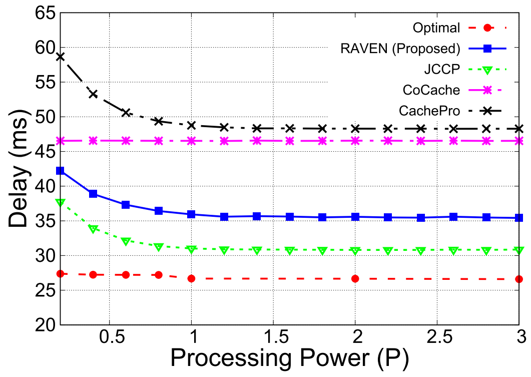
- Hit ratio - the fraction of requests fulfilled from the cache of the MEC servers.
- Average access delay - average time taken to download initial fragments of the video (sufficient to start the video playback) from any one of the MEC servers or CDN/content server to the user device.
- External traffic load - the amount of data fetched from CDN/content server to fulfill the user requests.

We compare the RAVEN caching method with the following approaches:

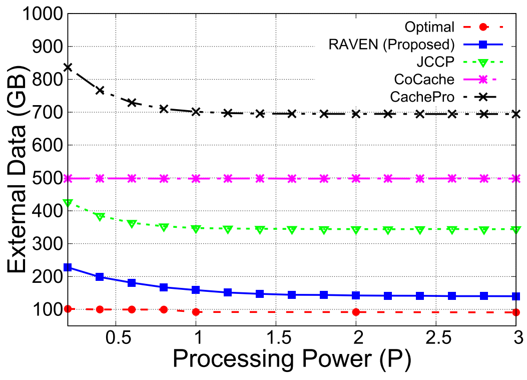
- Optimal - Optimal solution to the caching problem based on the ILP formulation for the given set of user requests. The solver knows all future requests to make the caching decision.
- CachePro - A caching and transcoding method for cellular RAN (Pedersen et al., 2016). In this work, the authors did not consider the collaboration among the MEC servers. Comparison of the results with this approach demonstrates the great merit of collaboration among MEC servers.
- CoCache - A collaborative caching policy without transcoding. Comparison of the results with this method is important to understand the effect of transcoding in a collaborative caching system.
- Joint Collaborative Caching and Processing (JCCP) - In JCCP, the authors proposed a joint collaborative caching and transcoding scheme (Tran et al., 2017b). In this work, MEC servers collaborate with each other to share the resources, and the video cached at one server can be served to the user who is connected to some other



(a) Hit Ratio



(b) Delay



(c) External Data

Fig. 5. Change in (a) hit ratio, (b) delay, and (c) external data with increase in processing power.

server. But in this work, the authors did not consider the cache consolidation and they deploy the simplistic Least Recently Used (LRU) cache replacement scheme. In contrast to their work, we come up with cache consolidation and also introduce a RAN aware cache replacement policy.

- Highest-bitrate video caching - In this case, only the highest bitrate of the video is cached on the MEC servers, and the request for lower bit-rate videos are served after transcoding the highest bit-rate version (Kumar and Vineeth, 2018).

The approaches are compared for different cache sizes, available processing powers at the MEC servers, and content popularity distribution. CachePro, CoCache, and JCCP algorithms perform a search operation in the cache for requested video and use the LRU algorithm for cache replacement. With an efficient implementation using hash tables, search operation and LRU algorithm execution can be performed in constant time ($\mathcal{O}(1)$). Therefore CachePro, CoCache, and JCCP have a constant time complexity ($\mathcal{O}(1)$). The proposed RAVEN method also provides a constant time complexity, which is equivalent to the time complexity of the stated algorithms.

5.1. Effect of change in request-arrival rate

To determine the scalability of caching only the highest bit-rate version of videos, we study the performance of the RAVEN method and highest bit-rate caching on different load conditions. To change the load, we reduce the inter-request arrival time (λ) in the simulation, which determines how often a user is sending the requests. Fig. 3 shows that in highest bit-rate video caching, the performance drops with the increase in load because it relies solely on the available processing power to serve the lower bit-rate users requests. In contrast, the change in the load does not affect the RAVEN, as it does not solely rely on the processing power of the MEC server to serve the users. The RAVEN method also beats the highest bit-rate caching in the lightly loaded condition ($\lambda = 50$). As caching the highest bit-rate video is not a scalable solution and well outmatched by RAVEN, we exclude the comparison with it hereafter.

5.2. Effect of change in cache size

Fig. 4(a) shows the change in hit-ratio on the increase in the cache size at a fixed processing power of 1 and displays that for cache size of 12%, the RAVEN algorithm outperforms the JCCP algorithm by 10% (for 10% of the cache) and performs close to the optimal solution. CachePro performs the worst since it does not have any collaboration among the MEC servers and hence the MEC servers could not fetch the content cached on other MEC servers. Fig. 4(b) depicts that the delay reduces with an increase in the cache capacity of MEC servers. For the cache size of less than 9%, RAVEN takes the minimum time to serve the users, and for higher cache size JCCP surpasses the RAVEN. JCCP replicates the content on different MEC servers and large cache size (9% cache size means that all four MEC servers can collectively cache 36%) quashes the performance impact of replication, leading to a lesser delay. On the contrary, RAVEN does not replicate the content to save the storage, and transfers the content from one MEC server to another to serve the user requests that induce some additional delay. For small cache size, RAVEN performs better (11.5% less delay for 5% cache size compared to JCCP) than other schemes. Likewise, in Fig. 4(c), downloaded data from the Internet decreases when the cache size is increased. RAVEN downloads 60% lesser data at cache size of 10% compared to JCCP and other schemes. Results indicate that CachePro performs the

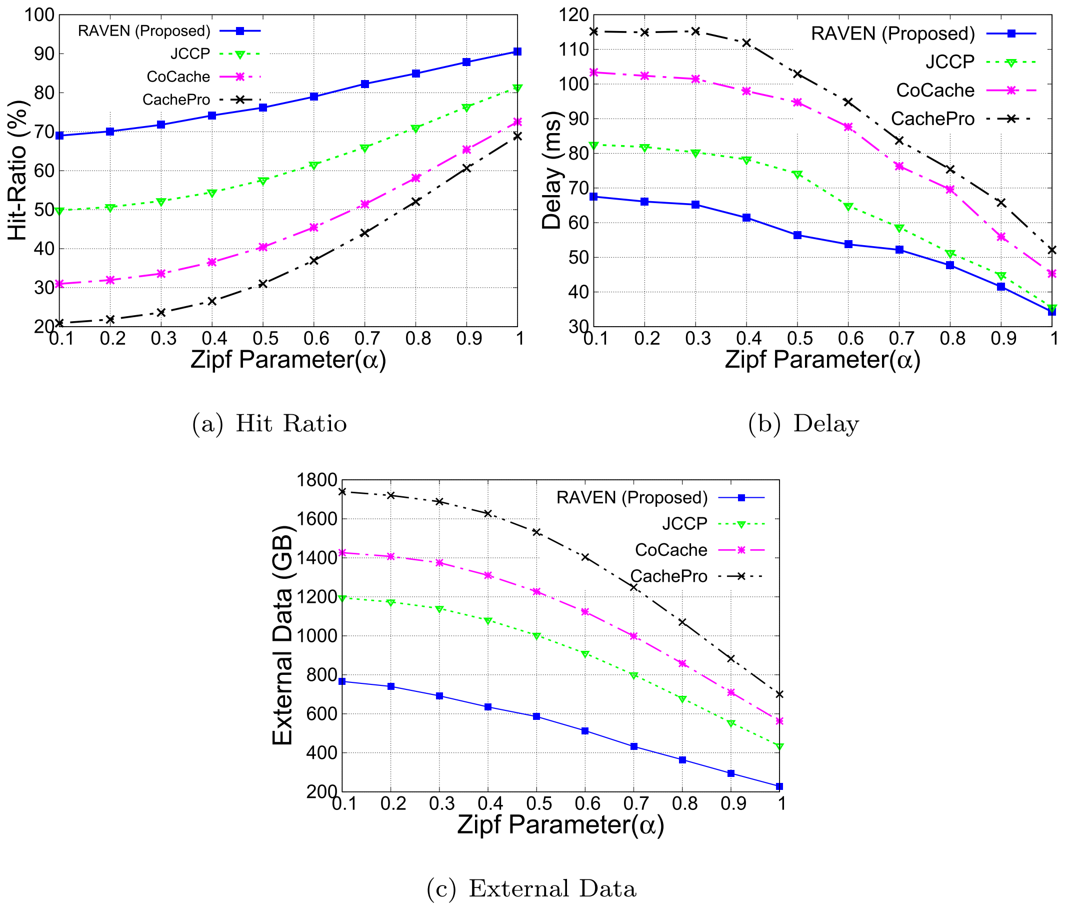


Fig. 6. Change in hit ratio, delay, and external data with change in skewness of video popularity distribution.

worst in all aspects, although it utilizes both processing and caching. On this basis, it is important to have collaboration among the MEC servers for efficient cache utilization. CoCache performs better than CachePro as it enables collaboration among the MEC servers, but it does not use the processing power for transcoding, thus it's outperformed by RAVEN and JCCP. For caching, JCCP practices both transcoding and collaboration, but because of content replication, it wastes the storage. In RAVEN, the cache is consolidated through collaboration, and it exploits the network information for cache replacement. Therefore, it attains the best results among the compared schemes in all aspects for small cache size.

5.3. Effect of change in processing power

Fig. 5(a) shows that the hit-ratio increases with an increase in processing power at a fixed cache size (12%). The RAVEN algorithm performs better than the JCCP and increases hit-rate by 8% for processing power of 1. The results in Fig. 5(b) illustrate the effect of the change in processing power on delay. The delay decreases with the increase in processing power. The RAVEN method has 13% more delay compare to JCCP with 1 processing power. RAVEN does not replicate the content, it thus achieves a better hit-ratio. For the same reason, it transfers the content from one MEC server to another to serve the users. Fig. 5(c) shows that the RAVEN downloads 54% less external data compared to JCCP on 1.2 processing power. Interestingly, the optimal solution does not exhibit any significant change with the change in processing power because it converges at around 12% of the cache size and does not improve any further. Results clearly establish that RAVEN outperforms JCCP and other schemes and gives a better performance for smaller cache and processing power. As CoCache does not transcode the cached videos, its results do not change with the change in processing power. However, it still performs better than CachePro, which employs the transcoding but does not have any collaboration among the MEC servers.

5.4. Effect of change in skewness of video popularity distribution

The proposed RAVEN method uses the video popularity distribution along with network information for caching decision, with this in mind, we evaluate it for different shapes of popularity distributions. To change the shape of underlying popularity distribution, we vary the value of Zipf parameter α in [0.1 1.0] that is used in video request generation. For the value of $\alpha = 1$, video popularity is very skewed and rank one video gets twice as much request as rank two videos, and $\alpha = 0$ means that all the videos will get the same number of requests following a uniform distribution. For the comparison, we assign 5% cache storage on each MEC server and $P = 1$ processing power. Results in Fig. 6 show that when the popularity skewness increases, the performance of all the schemes improves and their performance gap shrinks because most of the users will request only a few popular videos. Results in Fig. 6(a) display that the RAVEN algorithm raises the hit-ratio by 20% compared to other schemes for $\alpha = 0.1$ and 9% more hit-ratio of maximum skewed popularity. Fig. 6(b) depicts that RAVEN clearly outperforms the other schemes for $\alpha = 0.1$. However, for $\alpha = 1.0$, JCCP performs close to RAVEN, because JCCP replicates the popular content on different MEC servers and RAVEN instead transfers the content from one MEC server to another without replication. Furthermore, RAVEN saves 33% more data compared to JCCP for $\alpha = 0.1$ and 55% for $\alpha = 1.0$. Results in Fig. 6 confirm that for different video popularity distributions, the RAVEN method consistently outperforms the other schemes with a good

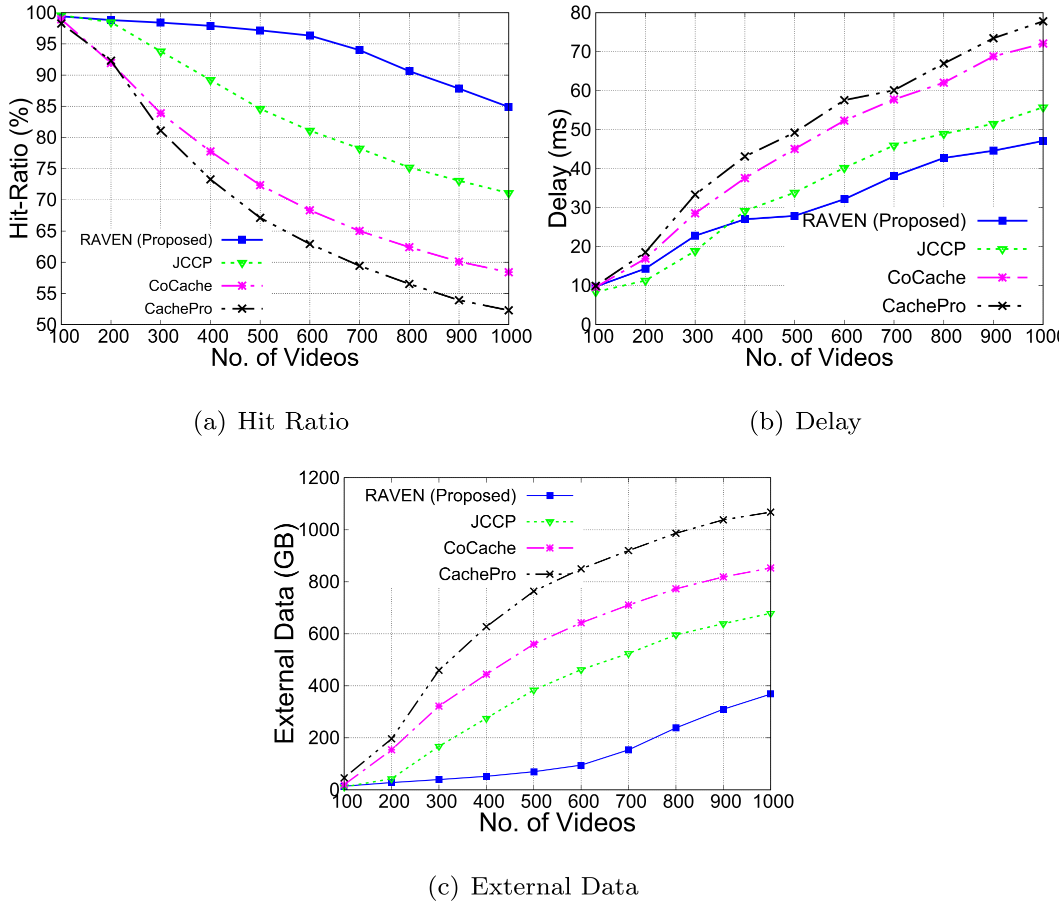
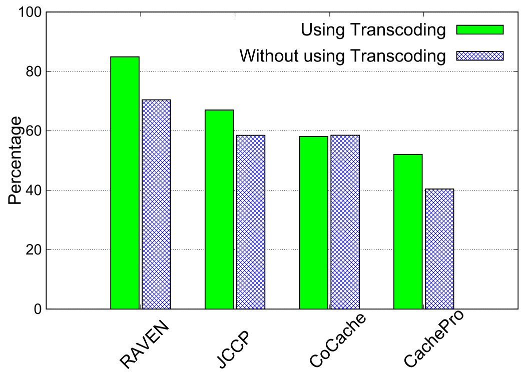
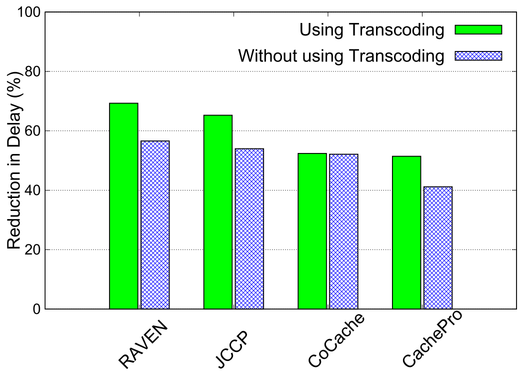


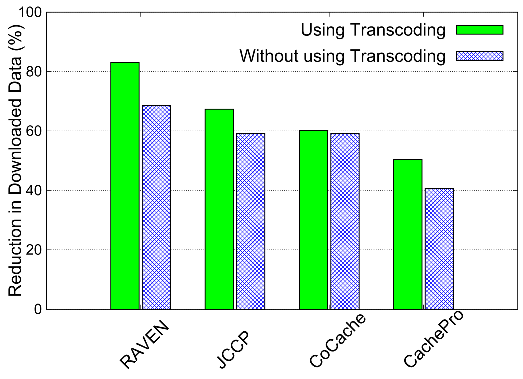
Fig. 7. Effect of change in number of videos in library on hit-rate, reduction in delay, and downloaded data.



(a) Hit Ratio



(b) Delay



(c) External Data

Fig. 8. Effect of the use of transcoding on hit-rate, reduction in delay, and downloaded data.

margin and it also attains the best results when video popularity distribution is uniform.

5.5. *Effect of change in video library size*

Here, we analyze the effect of video library size on the performance of the RAVEN. For this evaluation, we use Zipf parameter $a = 0.8$, a fixed 5% (of 1000 video library) cache size, and processing power $P = 1$. Results in Fig. 7 indicate that the performance of different methods gradually declines on the increase in the number of videos in the library. For a small number of videos (100), the performance of different methods is identical, which is reasonable because MEC can cache a significant part of the small video library. Therefore the proposed method's efficiency is not advantageous for a substantially small number of videos. An increase in the number of videos necessitates the efficient use of available cache storage, and it is evident from the performance difference in the results. RAVEN outperforms all the other methods for a significantly higher number of videos (400 or higher) across all the performance metrics.

5.6. *Video caching without transcoding*

To assess the impact of transcoding and benefits of the caching in the absence of it, we evaluate the proposed RAVEN and other methods when MEC does not perform the transcoding. Results in Fig. 8 points that when transcoding is engaged along with video caching, RAVEN attains an improvement of $\approx 14\%$ in hit ratio that is at least 4% higher than the performance gain by other methods. Therefore it is plausible to say that RAVEN employs computation power more diligently. Although transcoding has an impact on performance, RAVEN significantly outperforms the other methods in both cases (with or without using transcoding). Furthermore, RAVEN delivers a competing performance without using the transcoding as compared to the results achieved by other methods while utilizing the transcoding. CoCache does not use transcoding in video caching; thus, there is no noticeable difference in its performance with or without transcoding. Results in Fig. 8 reports that MNOs can benefit from video caching, even without using the transcoding, that can serve 70.43% of the users form the network edge and consequently decreases the average delay by 56.6% and reduces the download data from ISP by 68.5% which can notably cut the network load and OPEX.

6. Conclusion and future work

In this work, we design a RAN-aware adaptive video caching (RAVEN) method. RAVEN utilizes the radio network information provided by RNI API of MEC to select an appropriate bit-rates for video caching. We formulate the cache problem on MEC as an ILP to maximize the hit-ratio to serve most of the user requests from the network edge itself. As the optimization cannot be solved in real time, this work introduces a profit-based caching method to cache the video content on MEC servers. It utilizes the video popularity distribution and estimated video request bit-rates for caching decisions. RAVEN ensures the collaboration among the MEC servers, to avoid the replication, that has a positive impact on the cache performance. Further, RAVEN performs the transcoding to serve the different bit-rate requests from a cached video. Simulation results establish that the RAVEN outperforms the state-of-the-art caching schemes and performs close to the optimal solution as compare to others.

The introduction of Virtual Reality (VR) has brought new applications and use cases to the horizon. 360° video and VR are powerful techniques that offer the viewers an immersive experience and becoming increasingly popular. Caching a 360° video and AR/VR content is more challenging, compared to a regular video, because they demand a high bit-rate for streaming and comparatively huge storage space for caching. In addition, all the transmitted data is not consumed by the user, hence it presents an interesting problem, and we will explore it in our future work.

CRediT authorship contribution statement

Shashwat Kumar: Conceptualization, Methodology, Software, Writing - original draft. **Sai Vineeth Doddala:** Software. **A. Antony Franklin:** Supervision, Writing - review & editing. **Jiong Jin:** Supervision, Writing - review & editing.

Declaration of competing interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest, or non-financial interest in the subject matter or materials discussed in this manuscript.

Acknowledgements

This work was supported by the project "Low Latency Network Architecture and Protocols for 5G Systems and IoT", SERB, Govt. of India.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jnca.2020.102737.

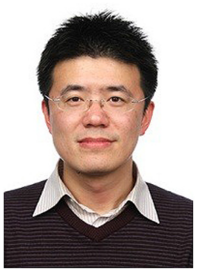
References

3GPP, TR36.814 v9.2.0: Evolved Universal Terrestrial Radio Access (E-UTRA); Further Advancements for E-UTRA Physical Layer Aspects.
Abdullahi, I., Arif, S., Hassan, S., 2015. Survey on caching approaches in Information centric networking. J. Netw. Comput. Appl. 56, 48–59, https://doi.org/10.1016/j.jnca.2015.06.011.
Bastug, E., et al., 2014. Living on the edge: the role of proactive caching in 5G wireless networks. IEEE Commun. Mag. 52 (8), 82–89.
Belshe, M., 2010. More bandwidth doesn't matter (much). www.belshe.com/2010/05/24/more-bandwidth-doesnt-matter-much/.
Bilal, K., Baccour, E., Erbad, A., Mohamed, A., Guizani, M., 2019. Collaborative joint caching and transcoding in mobile edge networks. J. Netw. Comput. Appl. 136, 86–99, https://doi.org/10.1016/j.jnca.2019.02.004.
Borst, S., Gupta, V., Walid, A., 2010. Distributed caching algorithms for content distribution networks. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9.
de Assuno, M.D., da Silva Veith, A., Buyya, R., 2018. Distributed data stream processing and edge computing: a survey on resource elasticity and future directions. J. Netw. Comput. Appl. 103, 1–17, https://doi.org/10.1016/j.jnca.2017.12.001.
ETSI, Mobile-Edge Computing - Introductory Technical White Paper.
ETSI, Mobile Edge Computing (MEC); MEC in 5G Networks.
ETSI, Mobile Edge Computing (MEC); Radio Network Information API.
ETSI, Mobile Edge Computing (MEC); Technical Requirements.
FFmpeg, https://www.ffmpeg.org/, (Accessed 5 November 2019).
Forecast, C.V., 2019. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022.
Gharaibeh, A., et al., 2016. A provably efficient online collaborative caching algorithm for multicell-coordinated systems. IEEE Trans. Mobile Comput. 15 (8), 1863–1876.
Hu, P., Dhelim, S., Ning, H., Qiu, T., 2017. Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. 98, 27–42, https://doi.org/10.1016/j.jnca.2017.09.002.
Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L., 2009. Networking named content. In: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT 09. ACM, pp. 1–12.
Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K.H., Shenker, S., Stoica, I., 2007. A data-oriented (and beyond) network architecture. ACM SIGCOMM Comput. Commun. Rev. 37 (4), 181–192.
Kumar, S., Vineeth, D.S., 2018. Edge assisted dash video caching mechanism for multi-access edge computing. In: 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1–6.
Kutscher, E.D., Eum, S., et al., July 2016. Information-centric networking (ICN) research challenges. RFC 7927.
Liu, J., Yang, Q., Simon, G., 2016. Optimal and practical algorithms for implementing wireless CDN based on base stations. In: 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring), pp. 1–5.
Mehrabi, A., Siekkinen, M., Yl-Jski, A., 2018. QoE-traffic optimization through collaborative edge caching in adaptive mobile video streaming. IEEE Access 6, 52261–52276, https://doi.org/10.1109/ACCESS.2018.2870855.
Ostovari, P., Wu, J., Khreishah, A., 2016. Efficient online collaborative caching in cellular networks with multiple base stations. In: 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 136–144.
Pedersen, H.A., et al., 2016. Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing. IEEE/ACM Trans. Netw. 24 (2), 996–1010.
Shanmugam, K., et al., 2013. Femtocaching: wireless content delivery through distributed caching helpers. IEEE Trans. Inf. Theor. 59 (12), 8402–8413.
Shen, B., et al., 2004. Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks. IEEE Trans. Multimed. 6 (2), 375–386.
Tran, T.X., Pompili, D., 2019. Adaptive bitrate video caching and processing in mobile-edge computing networks. IEEE Trans. Mobile Comput. 18 (9), 1965–1978, https://doi.org/10.1109/TMC.2018.2871147.
Tran, T.X., Hajisami, A., Pompili, D., 2017a. Cooperative hierarchical caching in 5G cloud radio access networks. IEEE Netw. 31 (4), 35–41.
Tran, T.X., Pandey, P., Hajisami, A., Pompili, D., 2017b. Collaborative multi-bitrate video caching and processing in mobile-edge computing networks. In: 2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS), pp. 165–172.
Wang, X., et al., 2014. Cache in the air: exploiting content caching and delivery techniques for 5G systems. IEEE Commun. Mag. 52 (2), 131–139.
Zhang, K., Leng, S., He, Y., Maharjan, S., Zhang, Y., 2018. Cooperative content caching in 5g networks with mobile edge computing. IEEE Wirel. Commun. 25 (3), 80–87, https://doi.org/10.1109/MWC.2018.1700303.
Zink, M., Suh, K., Gu, Y., Kurose, J., 2009. Characteristics of YouTube network traffic at a campus network measurements, models, and implications. Comput. Network. 53 (4), 501–514.

Shashwat Kumar received his B.Tech. degree in Information Technology from Uttar Pradesh Technical University, India, in 2011 and the M.Tech. degree in Computer Science and Engineering from the Sardar Vallabhbhai National Institute of Technology, Surat, India, in 2013. He is currently working toward his PhD degree in Computer Science and Engineering at the Indian Institute of Technology Hyderabad (IITH), India. His research interests include Multi-access Edge Computing (MEC), Video Caching, 360-degree Video Streaming, and Mobile Networks.

Sai Vineeth Doddala working in Samsung R&D Institute Bangalore as Senior Engineer in Wi-Fi & IOT team. He completed his B.Tech in the Department of Computer Science and Engineering from the Indian Institute of Technology Hyderabad, India. His research interests include Wireless Networks and Mobile Edge Computing.

Dr. Antony Franklin A received his B.E. degree in Electronics and Communication Engineering from Madurai Kamaraj University, India, in 2000 and M.E. degree in Computer Science and Engineering from Anna University, India, in 2002. He received his Ph.D. degree in Computer Science and Engineering from the Indian Institute of Technology Madras, India, in 2010. He is currently working as an Associate Professor at Indian Institute of Technology Hyderabad (IITH), India. Before joining IITH, he worked as a Senior Engineer at DMC R&D Center, Samsung Electronics, South Korea between 2012 and 2015 where he was involved in the development of 5G networking technologies. He also worked as a Research Engineer in Electronics and Telecommunications Research Institute (ETRI), South Korea between 2010 and 2012 where he was involved in Cognitive Radio Technology research. His current research interests include Mobile Networks, Cloud Radio Access Networks (C-RAN), Mobile Edge Computing (MEC), Internet of Things (IoT), SDN/NFV. He has published over 50 articles in refereed international journals and conferences. He is a senior member of IEEE and a member of ACM.



Jiong Jin (IEEE M’11) received the B.E. degree with First Class Honours in Computer Engineering from Nanyang Technological University, Singapore, in 2006, and the Ph.D. degree in Electrical and Electronic Engineering from the University of Melbourne, Australia, in 2011. He is currently a Senior Lecturer in the School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia. Prior to it, he was a Research Fellow in the Department of Electrical and Electronic Engineering at the University of Melbourne from 2011 to 2013. His research interests include network design and optimization, edge computing and distributed systems, robotics and automation, and cyber-physical systems and Internet of Things as well as their applications in smart manufacturing, smart transportation and smart cities.