applied sciences

MDPI

Article

A Novel Distributed Media Caching Technique for Seamless Video Streaming in Multi-Access Edge Computing Networks

Emmanuel Osei-Mensah [1,*], Saqr Khalil Saeed Thabet [1], Chunbo Luo [1,2], Emelia Asiedu-Ayeh [3], Olusola Bamisile [4], Isaac Osei Nyantakyi [1] and Humphrey Adun [5]

[1] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; saqr-thabet@std.uestc.edu.cn (S.K.S.T.); c.luo@exeter.ac.uk (C.L.); nyantakyiisaacosei@std.uestc.edu.cn (I.O.N.)
[2] Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou 313001, China
[3] School of Management Science and Economics, University of Electronic Science and Technology of China, Chengdu 611731, China; easieduayeh@std.uestc.edu.cn
[4] College of Nuclear Technology and Automation Engineering, Chengdu University of Technology, Chengdu 610059, China; boomfem@cdut.edu.cn
[5] Energy Systems Engineering Department, Faculty of Engineering, Cyprus International University, Haspolat-Lefkosa, Mersin 10, 99010, Nicosia 99258, Turkey; hadun@ciu.edu.tr
* Correspondence: ecoseimensah@std.uestc.edu.cn

Abstract: Online video is anticipated to be the largest fraction of all mobile network traffic aside from the huge processing tasks imposed on networks by the billions of IoT devices, causing unprecedented challenges to the current network architecture. Edge caching has been proposed as a highly promising technology to overcome this challenge by placing computational and data storage resources at the network edge to reduce latency and backhaul traffic. However, the edge resources are heavily constrained in their storage and computational capacities as large-scale deployments mean fairly distributing resources across the network. Addressing this limitation, we propose an edge video caching scheme that dynamically caches the first part of popularity-ranked video files on Multi-Edge Computing Access Node (MAN) servers envisioned to achieve higher cache hit ratios, lower latencies, and lower backhaul traffic. The concept of Regionally Organized Clouds (ROCs) with sufficient resources for file caching and compute-intensive tasks was introduced, and a formulation of the edge caching problem as an Integer Linear Programming (ILP) problem was made. Additionally, this study proposes a file view-time threshold for each cached video aimed at reducing the resource wastage caused when buffered contents are abandoned. Comparative evaluations of the proposed show its excellent performance over FIFO, Greedy, LFRU and TLRU schemes.

Keywords: Multi-Access Edge Computing (MEC); edge caching; distributed computing; resource allocation; edge network optimization

Citation: Osei-Mensah, E.; Thabet, S.K.S.; Luo, C.; Asiedu-Ayeh, E.; Bamisile, O.; Nyantakyi, I.O.; Adun, H. A Novel Distributed Media Caching Technique for Seamless Video Streaming in Multi-Access Edge Computing Networks. *Appl. Sci.* **2022**, *12*, 4205. https://doi.org/10.3390/app12094205

Academic Editor: Agostino Forestiero

Received: 24 March 2022
Accepted: 20 April 2022
Published: 21 April 2022

1. Introduction

Smartphone data traffic is projected to exceed PC data traffic in the next few years based on the Visual Networking Index by Cisco [1]. Internet video traffic accounts for the highest chunk of all smartphone traffic, reaching about 78 exabytes per month. The existing cellular networks will be congested; hence, service delivery will be deplorable if the challenges of video traffic are not addressed. Moreover, the current cloud architecture provides an inviable solution to the problem, as the number of connected edge-hosted containers is also estimated to be over 700 million to provide business resilience networking [2]. As a result of the heterogeneity of network devices and the dynamics of the network, video streams must be converted into multiple copies, each with different bitrates, and stored on main cloud servers. This approach incurs huge operational expenditure, in that it

necessitates greater storage capacities and processing power for the content housing and transcoding. Hence, there is a pressing demand to optimize current and future networks to facilitate the delivery of a high quality of service (QoS) and high quality of experience (QoE) at well under the operational expenditure. Network capacity expansion can alleviate the problem with congestion due to data surges, but it is not a feasible solution because it requires huge capital investment.

Recently, mobile edge computing, also known as the European Telecommunications Standards Institute (ETSI) Multi-Access Edge Computing (MEC), has been proposed as a promising solution and has been applied in many works in the literature [3–7]. The advent of this technology ushers in new ways of delivering better services to mobile users. The main challenge of the content delivery problem is that multimedia data is in data centers, while consumers are ubiquitous. The goal of every content delivery network (CDN) is to bring content closer to users via the best output performance considerations. Data centers are mostly located far from urban users. To realize a near-seamless QoE, in-network caching at the edge has gained great research momentum. ETSI MEC offers a network setting and cloud-computing abilities at the mobile network edge, empowering application, and content creators to launch new services, such as intelligent video acceleration, with low latency and high bandwidth. The closeness of storage and computing resources to users has been proven to lessen the burden on the core networks [7,8]. Caching popular and frequently accessed content within the network also reduces the transit service payments to Internet Service Providers (ISP), thereby reducing the total operational cost of the cellular network and achieving high QoE through fast content delivery by avoiding long-distance transmission [9]. MEC servers, however, are limited in two ways: (a) the caches in edge servers have scarcely constrained storage capacities; and (b) wireless network instability conditions lead to users requesting different bitrates of the same video, making the transcoding computationally resource expensive.

Further, caching all the multiple versions of the same video at the network edge poses a greater problem with the cache capacity. Hence, collaborative caching and processing have been introduced. The MEC server caches a higher bitrate version of a video which can be later transcoded into multiple lower versions to meet the demand of users. Clustered MEC servers collaborate to assist each other to deliver from their cache or help transcode the desired bitrate [10,11]. Some caching strategies, including first-in-first-out (FIFO), least-frequently used (LFU), and least-recently used (LRU) are basic, and their implementations are simple. However, their performance on cache hit ratio and latency is poor when considered in mobile edge caching for media content [12]. Other research-based strategies present solutions to gain efficient management of cache storage and short file delivery time for a less traffic communication [13]. Although proactive caching has been proposed to further reduce the file delivery time under constrained backhaul resources, it is strongly efficient when users have similar request patterns requiring accurate predictions of users' content demands with advanced learning algorithms [14]. In [15], rate-distortion characteristics of videos, video popularity, client's initial delay, and the transmission capacity for the base station and the MEC server were considered for each cache server to store only the best video presentations in an attempt to improve the clients' QoE.

To overcome the problems of video content management at mobile edge caching networks coupled with low-latency guaranteed transmissions, this study proposes a new distributed part of media (DPoM) caching approach that dynamically caches the first part of a video on a MEC Access Node (MAN). The edge cache placement problem to objectively maximize cache hit rate subject to storage and power constraints is modeled as an Integer Linear Programming (ILP) problem [16]. It is infeasible to obtain the optimum solution to the ILP for each user request in a time slot in real-time because it requires significant time due to its NP-completeness. Hence, a heuristic approach is designed to maximize the cache hit rate of the MAN servers in the likeness of a multiple knapsack problem [17], having the cache size of the MAN servers in a cluster to be the total capacity of each knapsack, the probability of a file being requested as the profit of the item, and the size of each file as the

weight of the item. The main idea of the proposed scheme is to cache all or most of the files at a regionally organized cloud (ROC). At the MEC access node (MAN), the cache storage is duo-partitioned: main cache and transient cache. The main cache stores the first part of the most popular files a content provider is serving at a period. Relative to the total length of a file, the first part of a file for caching can be 25% to 50% of the whole file since most users engage the video for a few minutes before abandoning the streaming session if they lose interest or experience long rebuffering sessions. The remaining parts of an ongoing streaming activity are fetched into the transient cache when a view-time threshold on the playing file is reached. This threshold is placed on a video to restrict the network to only delivering the remaining parts of a playing file once the threshold is reached, to curtail the high number of session abandonments caused by many factors such as longer startup delays, the negative role of unsteady bitrate changes, the negative role of the number of rebufferings, lack of sufficient interest in the content, etc. [18]. In this direction step, the transient cache is not unnecessarily filled with parts of files whose sessions have been abandoned. The fetched files in the transient cache are stored for a while to serve other users with similar requests before the whole cache is cleared during cache replacement to make room for other file parts in future streaming sessions. Cache update frequency policy of the main cache storage depends on users' preferences dynamism. Thus, our scheme does not only allow for more cached content at the MAN servers, but also improves the cache hit ratio, the latency associated with content fetching, and the overall backhaul traffic. The main contributions of this paper are as follows.

(1) This study introduces a resource-sufficient regionally organized cloud which supplies and organizes clusters of neighbor MAN cells.
(2) This study gives a formulation of the edge network caching problem as an Integer Linear Programming-based optimization problem objectively to maximize the cache hit ratio.
(3) This study sets a file view-time threshold on each file to minimize the backhaul traffic incurred when ongoing streaming sessions are abandoned.
(4) In this study, a design of heuristic algorithms to optimally place video contents dynamically in the clustered edge servers is computed.

The rest of the paper is sectioned as follows: We present related works in Section 2. In Section 3, we explain the caching system framework and formulate the caching problem. The proposed solution entailing the heuristic algorithms is detailed in Section 4. Section 5 presents the results and analyses based on extensive simulations. We conclude the paper in Section 6 with the final considerations and the future work.

2. Related Works

Recent video presentations with increasing user requests and the advancement of mobile technology have paved the way for new video applications and services. However, the resultant data traffic and network congestions define the limitations of current networks in massive data processing and handling. Mobile Edge Computing (MEC), particularly Edge caching, has been proposed to minimize network congestion and ensure low latency communication [19]. MEC is an approach to deploy micro cloud services at the edge of the network by offering storage and computation capabilities [20]. With the increasing demand for video streaming, several solutions have been proposed to cache the data at the network level closest to users [21–24]. However, MEC storage and computation resources are limited, as large-scale global deployment will seek a fair distribution of resources to output good overall system performance, meaning the resources should be utilized efficiently. Data offloading (caching) and task offloading (transcoding) have been the main hurdle addressed by the existing literature. In [25], Kumar et al. designed a RAN-aware adaptive video caching scheme that utilizes radio network information to select appropriate bit-rates for video caching considering video popularity distribution and estimated video request bit-rates from cached videos in a collaborative and replication-avoidable MEC network. Tran et al. [11] proposed a joint collaborative caching and processing framework that

supports adaptive bitrate video streaming. In [26], Dehghan et al. proposed a utility-driven in-network caching where contents are associated with a utility function corresponding to content's related hit probability in order to maximize the profits of caching. To ensure the effectiveness of the massively distributed but small-sized RAN caches, Ahlehagh et al. in [27] introduced RAN-aware reactive and proactive caching policies that utilize User Preference Profiles (UPPs) of all active users in a cell. Similarly, to offset the disadvantages of limited cache storage at the network edge, several learning approaches, incorporating content popularity prediction based on user preferences, clustering users based on similar content interests, and optimizing cache placement and replacement techniques have been proposed [7,28–30].

A combination of proactive prediction and replacement collaboration (PCR) among MECs to effectively manage the cache storage is proposed in [31]. To achieve high cache hit ratios, the authors in [32] proposed layered video caching for multiple social groups formed by mobile users based on their requests. A Stackelberg game model was developed to study the collaboration among multiple social groups and the cache node as users (players) compete with each other for the number of layers they request to cache. In [33], the authors proposed a lightweight, agile caching with a PID controller to efficiently control the rate for streaming high-quality data. The proposed algorithm minimizes the operations at the edge nodes to avoid overloading the highly constrained edge nodes. In [34], cooperative caching among MEC servers was considered and the video cache hit ratio was improved by caching multiple presentations of videos and transcoding the videos in real-time. The downside of this approach is that caching multiple versions of complete videos reduces the efficient utilization of the MEC cache space. To further improve the efficiency of the cache management, a Multi-Agent Reinforcement Learning (MARL)-based cooperative content caching policy has been proposed in [35], exploiting only the historical content demands of users when users' preferences are unknown.

Most of the aforementioned proposals either end up overburdening the edge servers with transcoding and machine learning tasks or use up cache storage quickly. In this situation, the responsiveness of the edge servers to guarantee seamless video streaming performance might be very poor in that the original purpose of the MEC is to ensure swift data processing of IoT data to provide low latency responses to devices. Video is massive data requiring reliable and high bandwidth connection for its delivery from servers to users. To reap the full benefits of MEC, content caching policies must efficiently cache and offload lightweight tasks to the edge servers. A more suitable solution involves clustering and cooperation among MEC servers, with at least one resource-sufficient node able to take heavy tasks on behalf of the other nodes [36,37]. Edge caching policies with effective edge resource allocation are key to maintaining a fine balance in computation and storage in the IoT Edge system. The following works [38–43] establish the edge caching problem as a resource-constrained optimization problem, decomposed into subproblems, and solved mostly with heuristic algorithms. The solutions are not always optimal, but evaluations show improvements over state-of-the-art caching policies considered in the works.

Worldwide attention has been drawn to the global carbon emission contribution of the information-communication technology (ICT) industry. Today, the Internet is the most needed and desired entity, connecting every facet of life and every sector of major economies, whose absence or malfunction causes heavy losses of monetary capital. The core and backhaul networks are often congested with ceaseless data traffic. Edge computing promises to reduce the over-dependence on the core and backhaul networks. Several studies have proposed edge caching policies that also capture techniques of further reducing the backhaul traffic [44–46]. However, the proposed schemes have not analyzed unused buffered content data during streaming session abandonments.

The existing approaches tackle the edge caching problem centrally, with content transcoding tasks offloaded to designated computing units without real-time balance in available computing resources in a dynamic and unpredictable wireless streaming environment. Additionally, no consideration has been given to the wastage of computing

and communication resources on unused content parts whenever streaming sessions are abandoned. Our proposed scheme addresses the problems of inefficient cache storage management and backhaul resource wastage, due to streaming session abandonments, exploiting a new method of content caching with strict file view duration threshold assignment, systematically designed to achieve higher cache hit ratios through the dynamic placement of more contents at the edge, and fewer content transmissions via the backhaul network.

3. System Framework

In this section, we present the framework for the proposed edge caching scheme. Figure 1 depicts the architecture of the proposed framework, comprised of several entities at different levels, where MAN servers are deployed in clusters, providing computation and storage resources to enable caching at the network edge. A MAN server can process requests directly from its local cache, neighbor MANs, ROC, or the central cloud.

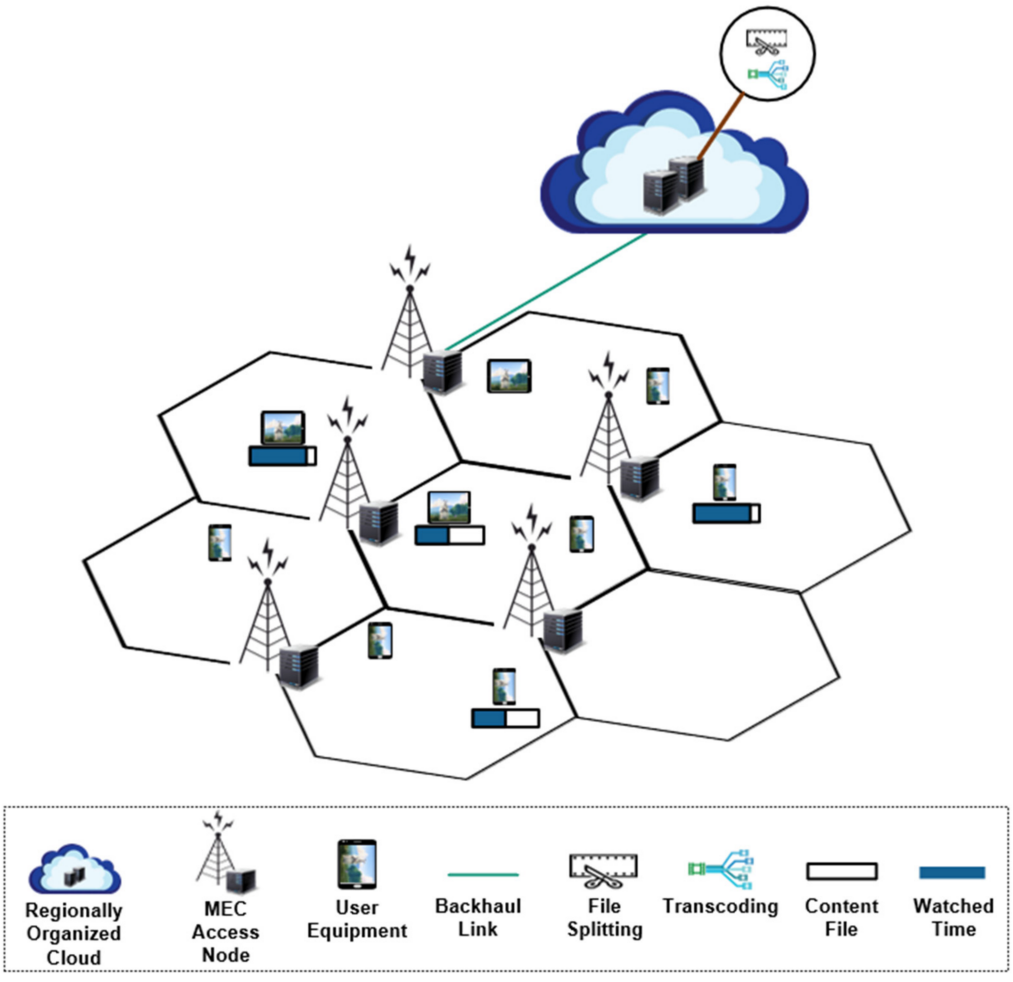

Figure 1. Edge caching scenario.

3.1. Entities for the Framework Design

1. **Central Cloud**: The central cloud connects to Regionally Organized Clouds via the core network. The facility houses all the media files of a content provider. It also serves data upon regional requests. When new content is published, the central cloud updates all cache catalogues on the regional clouds.
2. **Regionally Organized Cloud (ROC)**: A ROC connects to MEC access nodes via wireless or fiber backhaul. ROCs function as central clouds, but with fewer computing and storage capacities. A ROC caches content the region needs. It also computes

the popularity of content based on consumption data reports from the Edge Cache Manager. It correspondingly transcodes and serves the remaining parts of media content in the regional cache once an edge cluster manager requests for them. At the regional level of a cache network, users have similar preferences compared to those at a global level. The advantage realized is that a ROC can efficiently serve its region's needs, whiles following the region's data protection and rights requirements. Content providers can upload directly onto their ROCs to better serve their regional users.

3. **Edge Cluster Manager**: The edge cluster manager resides on a cluster's head MEC access node server at the edge of the network. Its primary function is to perform cache management locally. Users are served directly from the cache at a MEC access node if the requested file is in the cache. If the desired file is not in the cache, the edge cluster manager sends a cache search request to neighbor MEC edge nodes. It also directs user requests to ROCs directly for users to be served when files are not found in any of the clustered MANs.
4. **MEC Access Node (MAN)**: The MAN is the physical access point of the system. The user equipment (UE) connects to the entire network via a wireless or wired medium. Aside from functioning as connection media between user devices and the core network, MEC access nodes are equipped with both compute and storage capabilities. These are in the form of edge servers—transcoding servers and streaming servers. A transcoding server converts mostly higher resolutions and bitrates of video files to acceptable bitrates commensurate with users' network quality to realize jitter-free and shorter buffering times. A streaming server, however, fetches requested videos from the cache and serves them to users. If the video is not available at the MEC cache, the edge cluster manager redirects the streaming server to serve video streams from either neighbor MEC access nodes or ROCs. The MAN is also responsible for the provision of the wireless resource allocation strategy.
5. **User equipment**: Different users have different devices which are capable of streaming content over a wireless or wired network. Most users stream wirelessly via mobile devices such as smartphones, tablets, laptops. A small fraction of users still prefers wired connection streaming on devices such as desktop computers, laptops, smart TVs, and other smart home devices with screens.

3.2. Caching Strategy

We consider a multiple-layer caching network to deliver content from both the MANs and ROCs, which are equipped with cache storage capacities S_M and S_R respectively. We consider geographic regional locations with relating characteristics to be served by a ROC. For simplicity, a ROC has quite sufficient computing and storage resources ($S_R \gg \sum F_{max}\gamma$). Local content providers can upload directly to the regional platforms for users' consumption. Several MANs are directly linked to the ROC. We define a set of N MAN servers directly linked to a ROC as $M = \{m_1, \ldots m_n, \ldots m_N\}$ having main cache storage sizes $S_M = \{s_{m_1}, \ldots, s_{m_n}, \ldots s_{m_N}\}$ and utilized cache storage sizes $S_{M_U} = \{s^u_{m_1}, \ldots, s^u_{m_n}, \ldots s^u_{m_N}\}$. A set of K users with streaming equipment is defined as $U = \{u_1, u_2, \ldots, u_K\}$, and follows a two-dimensional Poisson distribution. A catalog of the V most popular video files dynamically cached at m_n is given by $F = \{f_1, f_2, \ldots, f_V\}$, where the popularity of a file is represented by the probability P_v that file f_v is requested by a user (r_{u_k}), following Zipf distribution [47], i.e., $P_v(f_v) = \frac{\Omega}{f_v{}^{\alpha}}$, where $\Omega = \left(\sum_{v=1}^{F} v^{-\alpha}\right)^{-1}$. The variable α characterizes the distribution. A higher α value indicates that a fraction of the content is more popular than the rest in the catalog, while a lower α value describes more consistent popularity among different fractions of the contents. Offline caching [22], in which content replacement sessions are carried out at off-peak times, is considered. Having fixed and insufficient cache storage sizes at the MANs, efficient content placement at the base stations is a determinant of the performance of the scheme. The wide variance of users' preferences requires caching many distinct files with similar requesting frequencies to realize a good caching scheme performance, i.e., record high cache hit ratios (H_c).

A more gainful technique is to cache only the first parts of the content files at the MAN servers, as depicted in Figure 2. The remaining parts are stored on the ROC server with a delivery initialization condition: when the view-time threshold λ_v of a file is reached. A user u_k requests for video f_v of size s_{f_v} (small file $SF \leq A$ MB; medium file A MB $< MF \leq B$ GB; large file $LF > C$ GB, where A, B, *and* C are file sizes determined by the content provider such that $A < B < C$) with quality q_{f_v} at MAN server m_n. The variable $c_{m_n}^{q_{fv}} \in \{0,1\}$: $n \in N$, $v \in V$ indicates if a video file f_v with quality q is cached on a MAN server n. If the sought-after file is in the cache, $c_{m_n}^{q_{fv}} = 1$, otherwise $c_{m_n}^{q_{fv}} = 0$. To determine if a request is served from the local MAN server, any neighbor MAN server in the cluster, or a ROC, with or without transcoding, we define ternary variables φ, ψ, and γ. If a user's video request for f_v in quality q is served by the local MAN, then $\varphi_{m_n}^{q_{fv}} = 1$, and 0 otherwise. If the file is not cached locally but it can be served by any of the neighbor cells then $]\varphi_{m_k}^{q_{fv}} = 1$, and $\varphi_{m_k}^{q_{fv}} = 0$ indicates that the file is not in any neighbor MAN's cache storage. In this case, the file must be served from the ROC's cache, $\gamma_{m_n}^{q_{fv}} = 1$. Since ROCs have sufficiently high cache capacities and computational resources, we overlook the rare situation where a requested file is unavailable on the ROC server ($\gamma = 0$). We denote $\psi_{m_n}^{\hat{q}_{fv}} = 1$ for a local MAN server transcoding video f_v from quality $Q \to \hat{q}$. Likewise, if any k-th MAN server in the cluster of neighbor cells transcodes a video f_v from quality $Q \to \hat{q}$ which satisfies a user's request, then $\psi_{m_k}^{\hat{q}_{fv}} = 1$. Consequently, let $W_{m_n}^{q_{fv}} = \left(\varphi_{m_n}^{q_{fv}} + \psi_{m_n}^{\hat{q}_{fv}}\right)$ and $W_{m_k}^{q_{fv}} = \sum_{n \neq k}(\varphi_{m_k}^{q_{fv}} + \psi_{m_k}^{\hat{q}_{fv}})$, where $W_{m_n}^{q_{fv}}$ is the state that a user's request is served by the local MAN server and $W_{m_k}^{q_{fv}}$ is the state that a user's request is served by any of the MAN servers in the cluster other than the local MAN server of the user. The cache network satisfies the request of a user by following one of the three stated cases, which results in (1).

$$W_{m_n}^{q_{fv}} + W_{m_k}^{q_{fv}} + \gamma_{m_n}^{q_{fv}} = 1;\ \forall n, k \in N, f \in F \tag{1}$$

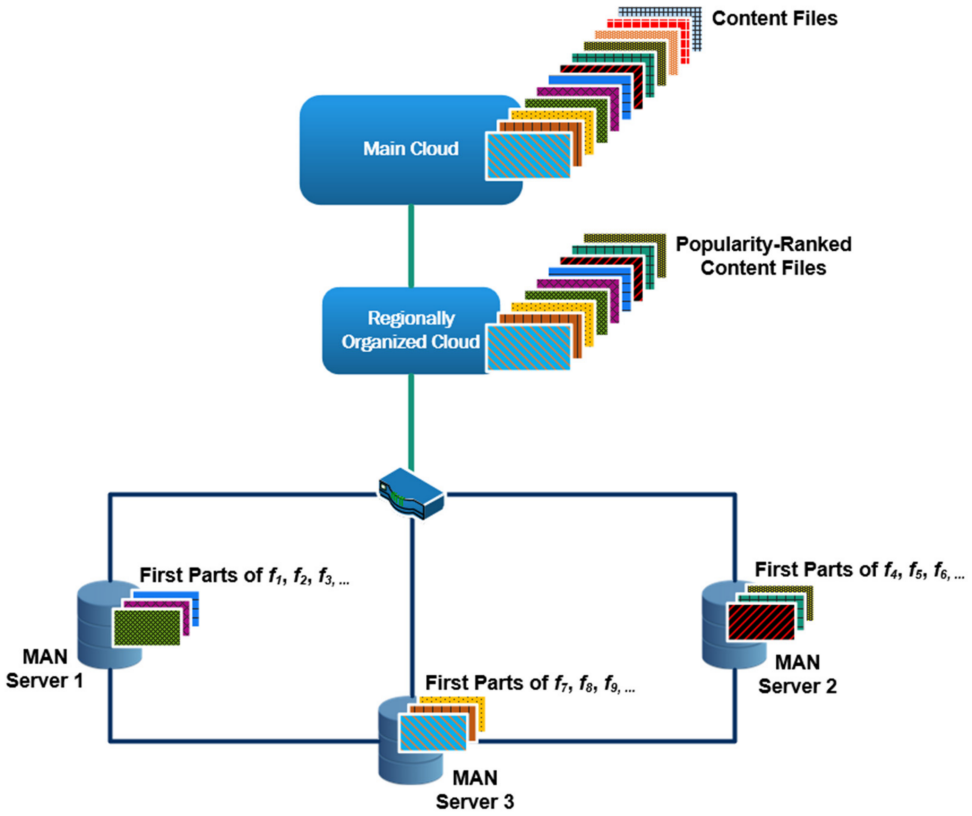

Figure 2. Cache content management.

We intend to serve all or most of the users' requests from the local or neighbor MAN servers to maximize the H_c. A request served by a ROC is regarded as a cache miss considering the constraint of user QoE which can be low for high latencies between the MAN servers and the ROC. The cache hit (H_C) formulation is given as

$$H_C = \sum_{n \in N} \sum_{q_{f_v} \in R_n} (W_{m_n}^{q_{f_v}} + W_{m_k}^{q_{f_v}}) \tag{2}$$

3.3. Problem Formulation

In this section, the formulation of the video caching problem as a cache hit rate maximization under the cache size of the multiple MAN servers and their processing power constraints is presented.

3.3.1. Problem 1

The objective function in (3) represents the total number of cache hits of the edge cache network. In (4) and (5), the decision variables ($c_{m_n}^{q_{fv}}$ and $c_{m_k}^{q_{fv}}$) are constrained to 1 only if f_v is cached on the MAN server n or k respectively. The constraints in (6) and (7) set $\psi_{m_n}^{\hat{q}_{fv}}$ and $\psi_{m_k}^{\hat{q}_{fv}}$ to one when the file f_v is cached in a higher bitrate presentation $\hat{q}$ which requires transcoding to the requested bitrate presentation q. The constraint in (8) allows content to be fetched from only one place at a time. In (9), a cache capacity violation constraint is set for the system. An upper bound on the consumed processing power needed to transcode a video from a higher presentation to a requested one is set in (10).

$$\text{Maximize } H_C = \sum_{n \in N} \sum_{q_{f_v} \in R_n} (W_{m_n}^{q_{f_v}} + W_{m_k}^{q_{f_v}}) \tag{3}$$

$$\text{Subject to } \varphi_{m_n}^{q_{f_v}} \leq c_{m_n}^{q_{fv}}, \ \forall n \in N, \ f \in F, \ q \in Q \tag{4}$$

$$\varphi_{m_k}^{q_{f_v}} \leq c_{m_k}^{q_{fv}}, \forall k \in N, \ f \in F, \ q \in Q \tag{5}$$

$$\psi_{m_n}^{\hat{q}_{f_v}} \leq min\left(1, \sum_{\hat{q}=q+1}^{Q} c_{m_n}^{q_{fv}}\right), \forall n \in N, \ f \in F, \ q \in Q \tag{6}$$

$$\psi_{m_k}^{\hat{q}_{f_v}} \leq min\left(1, \sum_{\hat{q}=q+1}^{Q} c_{m_k}^{q_{fv}}\right), \ \forall k \in N, \ f \in F, \ q \in Q \tag{7}$$

$$W_{m_n}^{q_{f_v}} + W_{m_k}^{q_{f_v}} + \gamma_{m_n}^{q_{f_v}} = 1, \ \forall n \in N, n \neq k, \ f \in F, \ q \in Q \tag{8}$$

$$\sum_{q_{f_v} \in R_n} s_{f_v} c_{m_n}^{q_{fv}} \leq s_{m_n}, \ \forall n \in N, \ f \in F, \ q \in Q \tag{9}$$

$$\sum_{q_{f_v} \in R_n} P_{\hat{q} \to q} \left(\psi_{m_n}^{\hat{q}_{f_v}} + \sum_{k \neq n} \psi_{m_k}^{\hat{q}_{f_v}}\right) \leq P_n^u, \ \forall n \in N, n \neq k, \ f \in F, \ q \in Q, \ u \in U \tag{10}$$

3.3.2. Problem 2

Further, we find an appropriate file consumption threshold λ_v for each video file. The file consumption threshold parameter allows the cache manager to specify a content view duration threshold $f_{WT_v} \geq \lambda_v$ which initializes the fetching of the remaining parts of a file from the ROC server and temporarily caches on the transient cache of the local MAN server. Hence, to ensure smooth uninterrupted streaming, the necessary condition is for a user to continuously engage with the selected video until the set duration threshold is reached. We propose this approach to reduce the wastage of the backhaul bandwidth and the MAN cache storage scenarios of streaming session abandonments by users. The variable $d_{m_n}^{q_{fv}} \in \{0, 1\}$ indicates whether the remaining parts of the requested video f_v in quality q

and cached at MAN server n should be transmitted. The condition of reaching the threshold, i.e., $f_{WT_v} \geq \lambda_v$ gives $d_{m_n}^{q_{fv}} = 1$, whereas unsatisfied threshold condition ($f_{WT_v} < \lambda_v$) gives $d_{m_n}^{q_{fv}} = 0$. Assuming the backhaul bandwidth is divided into β subcarriers and each subcarrier is shared by multiple users in a time-division manner, then the transmission rate R for the k-th user in the n-th cell on the m-th subchannel is given by Equation (11), where $\alpha_{k,n}^m$ denotes the time-sharing factor of user k in the MAN cell n on subcarrier m, $\chi_{k,n}^m$ is the channel access indicator (the decision variable). If the n-th MAN server serves the k-th user, $\chi_{k,n}^m = 1$, and zero otherwise.

Minimize

$$R_{k,n}^m = \sum_{m=1}^{M} \alpha_{k,n}^m \frac{B}{N} \log_2 \left(1 + \frac{\chi_{k,n}^m \cdot SINR_{k,n}^m}{\alpha_{k,n}^m \frac{B}{N}} \right) \tag{11}$$

Subject to

$$f_{WT_v} \geq \lambda_v, \ \forall f \in F \tag{12}$$

$$\sum_{k \in K,\ n \in N} \chi_{k,n}^m = 1, \ \ \forall n \in N, n \neq k \tag{13}$$

The second optimization objective is to maximize the efficient utilization of the backhaul resources by minimizing the content transmission rate when the view threshold is not reached. In (11), the objective function is to minimize a user's channel utility. The constraint in (12) controls the decision variable $d_{m_n}^{q_{fv}}$, which instantiates the final channel access indicator variable $\chi_{k,n}^m$. The constraint in (13) sets all channel access indicators to one when all the view-time thresholds are met. The formulated problems are NP-complete ILP problems, hence obtaining an optimal solution in polynomial time is infeasible for the reason that in a clustered cooperative environment, knowledge of all the possible video requests is required to solve the ILP and that is unattainable. Therefore, to achieve our objective of maximizing the cache hit rate of the edge caching network, we design a distributed part of media (DPoM) caching method which follows a multiple knapsack optimization, considering the MAN servers in a cluster as knapsacks with capacities equivalent to the cache sizes of the MAN servers, the profit of caching, i.e., users served from the edge, as the item's value, and the cache size requirements for each content file as the weights of the items. Heuristically, the method randomizes the selection of cacheable contents from the entire catalog by dynamically caching the first parts of contents having maximum popularity at every content drawing.

4. Proposed Solution

4.1. Distributed Part of Media (DPoM) Management

Content placement and caching at the network edge are carried out intelligently using a dynamic and distributive technique. The first part of the solution to the edge caching problem is presented in Algorithm 1, which depicts the steps for file splitting and placement. Firstly, DPoM performs content popularity comparison for any two randomly drawn files for all cacheable contents, then checks the size suitability for caching. For a file in *SF* range, there is no need for file splitting, so the entire file is placed in the main cache of a MAN server. For a file of size in the *MF* range, the splitter outputs two parts with the first part placed in the main cache. Similarly, a file in the *LF* range is split into multiple parts and the first part is cached. All remaining parts of split files are stored on the ROC for faster retrieval to the transient caches when users continuously engage their requested contents from the MAN servers. The MAN servers are represented as a set of knapsacks $M = \{1, \ldots, m\}$ with capacities $S_M = \{s_{m_1}, \ldots, s_{m_n}, \ldots s_{m_N}\}$, and the video contents as the set of n items, each having both profit of caching which is the request probability and weight (cost) which is the size of the item. Each item has a size which is divisible. The problem is to find the number of items to be put in each knapsack such that: (a) the total value of the assigned items is maximum; (b) the total size of items assigned does not exceed

the capacity of the knapsack; and (c) the total number of the assigned items does not exceed the upper bound. Hence, the complexity of the proposed algorithm is $O(n^2 + nm)$.

Algorithm 1 DPoM File Splitting and Placement Algorithm

Input : New videos $F = \{f_1, f_2, \ldots, f_V\}$
Output: Optimal content placement on clustered MAN servers
Initialize Caches on MAN servers $S_M = \{s_{m_1}, \ldots, s_{m_n}, \ldots s_{m_N}\}$
    while $s^u_{m_n} \leq s_{m_n}$ draw files for popularity comparator
        for $f_v.popularity() == f_{v+1}.popularity()$?
            if $f_v.popularity() > f_{v+1}.popularity()$ do
            switch(expression)
                begin
                    case $s_{f_v} \leq SF$ then
                        No file splitting required : Cache f_v on MAN server n
                        break
                    case $SF < s_{f_v} \leq MF$ then
                        Split f_v into f^1_v and f^2_v. Cache only f^1_v on MAN server n
                        break
                    case $MF < s_{f_v} \leq LF$ then
                        Split f_v into h parts for each $h \geq X$ GB. Cache only f^1_v in quality q on MAN server n
                        break
                end
            end if
        end for
    end while

4.2. DPoM File Fetching Algorithm

The process of responding to user content requests is detailed in Algorithm 2. The algorithm initializes the MAN servers for content fetching and processing power for transcoding. At the request for a video content file f_v in quality q by user u_k, DPoM first checks if the file is on the local MAN server n and serves u_k if it is available ($c^{q_{fv}}_{m_n} = 1$). If the desired video is not in cache at the local MAN server, the edge cache manager searches cooperatively with the other MAN servers and serves the user u_k if the file f_v is cached on any k-th neighbor MAN server ($c^{q_{fv}}_{m_k}$=1; $k \neq n$). To efficiently utilize the limited processing power of the MAN servers, if the requested quality q is within the cluster, there is no need for transcoding. However, if there exists f_v in a higher quality $\hat{q}$ in the local n-th or alternate k-th MAN server, the file is transcoded to the desired quality and served to the user ($\psi^{\hat{q}_{fv}}_{m_n} = 1$; $\psi^{\hat{q}_{fv}}_{m_k} = 1$). If the video is not on any of the servers clustered as neighbors, the algorithm presents a search request to the ROC.

Since the ROC has a sufficiently large storage capacity, there is a higher probability that the file will ultimately be served from its server ($\gamma^{q_{fv}}_{m_n} = 1$). However, the central cloud is the last resort if the desired file is unavailable on the ROC server. The ROC always transcodes the requested file to the desired quality before forwarding it to the MAN server. Once a user u_k reaches the view-time threshold of the file f_v, the remaining parts are transcoded by the ROC and sent to the transient cache of the user's local MAN server ($f_{WT_v} \geq \lambda_v$).

Algorithm 2 DPoM Fetching Algorithm

Input : File request (f_v) by a user u_k.
Output : Cache Hit Rate (H_C), Latency (L_T), and Backhaul Traffic (T_B)
1 : Initialize : Available processing power P_n^u
2 : Initialize : Cache on MAN servers $s_{m_1}, \ldots, s_{m_N}$
3 : for $n \in 1, \ldots, N$ do
4 : for each request r_{u_k} do
5 : if $c_{m_n}^{q_{fv}} == 1$ then
6 : Serve u_k from MAN server n
7 : else if $c_{m_k}^{q_{fv}} == 1; k \neq j$ then
8 : Fetch f_v from MAN server k and serve u_k
9: switch (expression)
10: begin
11 : case $\gamma_{m_n}^{q_{fv}} == 1$ then
12 : Transcode from $\hat{q}$ to q and send f_v from the ROC server to the MAN server n
13: break
14 : case $\psi_{m_n}^{\hat{q}_{fv}} == 1$ and $P_{\hat{q}\to q} \leq P_n^u$; $\hat{q} > q$ then
15 : Transcode f_v from $\hat{q}$ to q on MAN server n
16: break
17 : case $\psi_{m_k}^{\hat{q}_{fv}} == 1$ and $P_{\hat{q}\to q} \leq P_k^u; k \neq j$ then
18 : Fetch f_v after transcoding from $\hat{q}$ to q on MAN server k
19: break
20: end
21 : if $s_{f_v} \leq SF$ then
21: Complete fetching. Record $Requested.(Amount) = 1$; $H_C = 1$
22 : else if $SF < s_{f_v} \leq MF$ and $f_{WT_v} \geq \lambda_v$ then
23 : Send f_v^2. Complete fetching. Record $Requested.(Amount) = 1$; $H_C = 1$
24 : else if $MF < s_{f_v} \leq LF$ and $f_{WT_v} \geq \lambda_v$ then
25 : Fetch remaining parts of f_v. Complete fetching. Record $Requested.(Amount) = 1$; $H_C = 1$
26: end if
27: end for
28: end for

5. Experimental Evaluation

5.1. Experimental Setup

The efficiency of the proposed edge caching scheme has been raptly evaluated using the Java-based simulator [48] under benchmarked simulation parameters as presented in Table 1. We created a cluster of five MEC access nodes with cell coverage size of 200 m. Each MEC access node has 20 channels with a channel bandwidth of 20 MHz and a transmitting power of 40 W. Xender is one of the world's leading applications for file transfers and sharing with the convenience to transfer files of different types and sizes without cellular internet connection, or cables. We use Xender's captured video tracking for the month of August in 2016 as recorded in [29], with 153,482 video deliveries out of 271,785,952 video requests from 450,786 mobile users [49]. Most of the video contents are small files under 200 MB and the tracking can be fitted with the Mandelbrot-Zipf (MZipf) distribution of platform factor -0.88 and skewness factor of 0.35. The MZipf value gives an open relation of content popularity; however, this can be easily adjusted to realize a closer popularity among contents during simulation setup. When the Zipf distribution is set low, the cached contents are closely ranked in popularity making every one of them highly probable to be requested.

Table 1. Simulation parameters.

Parameter	Value
No. of clustered MAN servers	5
Cell coverage size	200 m
No. of content files (f_V)	10,000
MAN to MAN delay	20 ms
ROC to MAN delay	100 ms
Zipf's parameter (α)	0.55
Request rate	0.4~1.0
No. of active users	100~300
Processing Power (P_n^u)	1
Transmission power	46 dBm
Channel bandwidth	20 MHz
Noise power	−95 dBm
Cache size	1~20 GB
Number of iterations	100

To determine the superiority of the proposed caching scheme, different set of experiments have been performed. The first experiment is carried out by keeping all parameters constant and varying only the cache size of the MAN servers. A hundred iterations are run, and the results are averaged. The relationship between the varying cache sizes and key performance metrics are drawn with conclusions in the results and discussion section. Similarly, for the second to the fifth experiments, the file request rates, the number of users, the Zipf distribution, and the number of files are respectively varied while keeping all the other parameters set at the constant values depicted in Table 1. We evaluated the proposed scheme in comparison with four other baseline schemes with every experiment scenario set to run at a hundred iterations and averaging the results. With an exception to the LFRU and TLRU, the rest of the baseline algorithms have been natively implemented as part of the simulator. The LFRU and TLRU are modified versions of the LFU and LRU algorithms which have also been natively designed.

Running a scenario begins with the simulator initializing the size and popularity of all contents. Time slices were created to balance cache replacements, file request arrivals, and results logging. After each complete run of a time slice, the selected algorithm adjusts the cache in the MAN servers. The requests in each time slice arrive and the cache hit rate, latency, and backhaul traffic are recorded.

5.2. Performance Metrics

Performance measurements for analytical evaluations of the schemes involve cache hit rate, latency (delay), and network backhaul traffic. Key parameters like the cache size of the MAN servers, number of active users, number of video requests, and transmission power are varied to observe the correlation of the edge cache network performance and varying parameters.

5.2.1. Cache Hit Rate

The cache hit ratio is the total number of content requests which are successfully served by the local MAN server or a neighbor MAN server in a streaming period. We intend to serve user requests from the MAN servers. Any request which is unserved by the clustered servers is deemed a cache miss. Equation (2) can be simplified as

$$Cache\ Hit\ Rate = \frac{Total\ Hit\ Count\ in\ a\ time\ slice}{Total\ Request\ in\ a\ time\ slice}$$

5.2.2. Latency

Latency, also known as the network delay in fetching content, is the time taken for a user request to be served. MAN to MAN latency is set to 20 ms to test the efficiency of the schemes in worse scenarios.

$$Latency = \frac{Total\ Cache\ Miss \times 10}{Queue\ size\ of\ all\ users\ in\ time\ stream}$$

5.2.3. Backhaul Traffic

The backhaul traffic is the total size of all transmitted contents plus the total requested contents that could not be served by the clustered MAN servers. If a content request scenario necessitates content transfers from neighbor MAN servers or the ROC, the size of the data transferred is also recorded as the backhaul traffic.

$$Backhaul\ Traffic = Total\ size\ of\ data\ transmitted$$

5.3. Baseline Schemes

- Greedy Caching: This approach adopts a greedy approach that considers cache hit rate maximization at each caching level by making caching decisions based on cache miss stream from downstream caches [50]. There are several variations, but the main objective is to replace the cached content with the lowest cache utility. The motivation of the greedy algorithm is to use caching cost to determine which of the same-size contents should be cached. When a cached content is fetched, a value is assigned to it, i.e., the cost of bringing the content to the cache-store. The eviction policy replaces the content with the minimum value and then all the cached contents reduce their assigned values by the minimum value. The time complexity for this implementation is $O((n-1) \times \log(n))$.
- First In, First Out (FIFO): Without regard for the recency or frequency of accessibility of contents, the FIFO scheme evicts the contents in the order they were added. The first contents added during cache replacement are evicted first. In a FIFO queue, cached contents are placed in the tail of the queue. During cache replacement, cached contents are moved from the head until there is enough space for new contents. The time complexity of the FIFO algorithm is $O(1)$.
- Time-aware Least Recently Used (TLRU) [51]: TLRU is a variant of LRU and most suitable for information-centric networking (ICN), content delivery networks (CDNs), and distributed networks in general. A time to use (TTU) term is introduced to time-stamp every cached content, which stipulates the usability time for the contents based on the locality of the content and content provider notice. TLRU guarantees that less popular and low lifetime content should be replaced with the incoming content. TLRU has a time complexity of $O(1)$.
- Least Recently Used (LFRU) [52]: The benefits of LFU and LRU schemes are realized in this combination of caching schemes. It is also suitable for ICN and CDNs. To address the drawback of the Least Frequently Used (LFU) policy where multiple contents are at par on their frequency of file requests, the LFRU is more suitable to distinguish contents on a more elaborate scale, using both frequency and recency of the content requests. Each cacheable content is assigned a Combined Recency and Frequency (CRF) value. The content with the lowest CRF value during cache replacement is evicted. Each file request of a cached content increases its CRF. The time complexity of implementing LFRU ranges from $O(1)$ to $O(\log_2 n)$.

5.4. Results and Discussion

5.4.1. Performance Evaluation under Cache Size

The relationship of the cache size variation on cache hit rate, latency, and backhaul traffic is depicted in Figure 3. The cache hit ratio significantly increases with increasing

cache size for all schemes. The proposed scheme outperforms the other schemes by small margin gains. This shows that an increase in the cache size of MAN servers improves the cache hit ratio of all caching schemes.

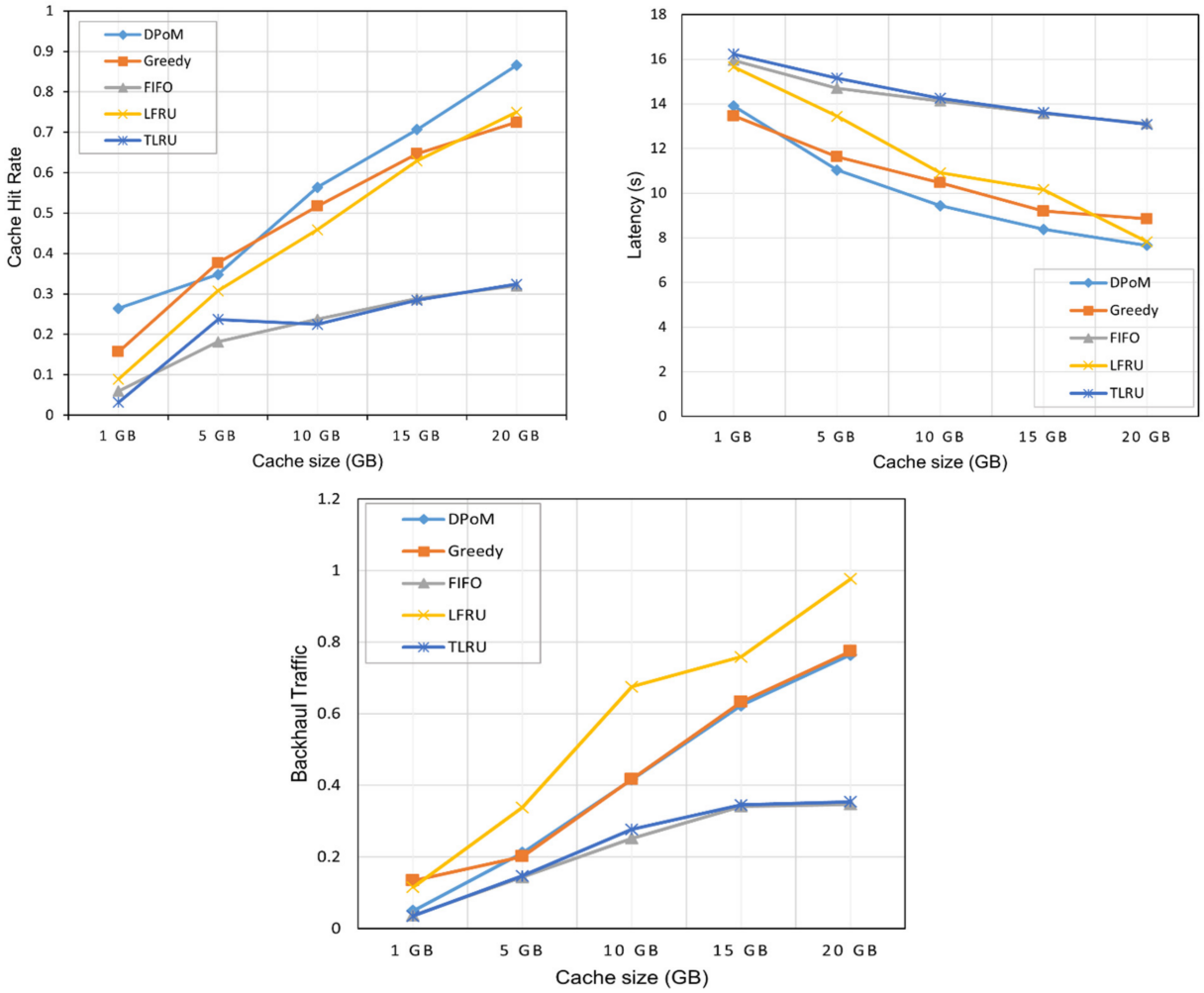

Figure 3. Performance of cache size variation on hit rate, latency, and backhaul traffic.

The second sub-figure in Figure 3 depicts the latency performance relationship with increasing cache size. Similar to the hit rate performance, the proposed is the best scheme in this comparison. However, the performance of the LFRU and Greedy schemes comes close to that of the proposed scheme. The increase of the cache size had a significant impact on the network latency, as larger cache storages allow for the placement of many content files thereby decreasing the file fetching time.

An increase in cache size also increases the backhaul traffic, as depicted in the third sub-figure of Figure 3. In this performance comparison, DPoM shows an average performance, better than the LFRU, at par with Greedy, and lagging behind TLRU and FIFO. Since DPoM parts the files into several parts, with most files in the small files range, the view duration thresholds of many files are easily reached, and hence there are more file transmissions over the backhaul network.

5.4.2. Performance Evaluation under Request Rates

The request rate indicates the frequency at which content files are requested by users. This parameter is essential in measuring the cache network's responsiveness in serving all user requests. In this evaluation, depicted in the first sub-figure of Figure 4, the proposed scheme shows a steady performance with increasing request rates. It performs better than

all the baseline schemes. Conclusively, an increase in the request rates has no negative impact on the scheme's ability to achieve good hit rates.

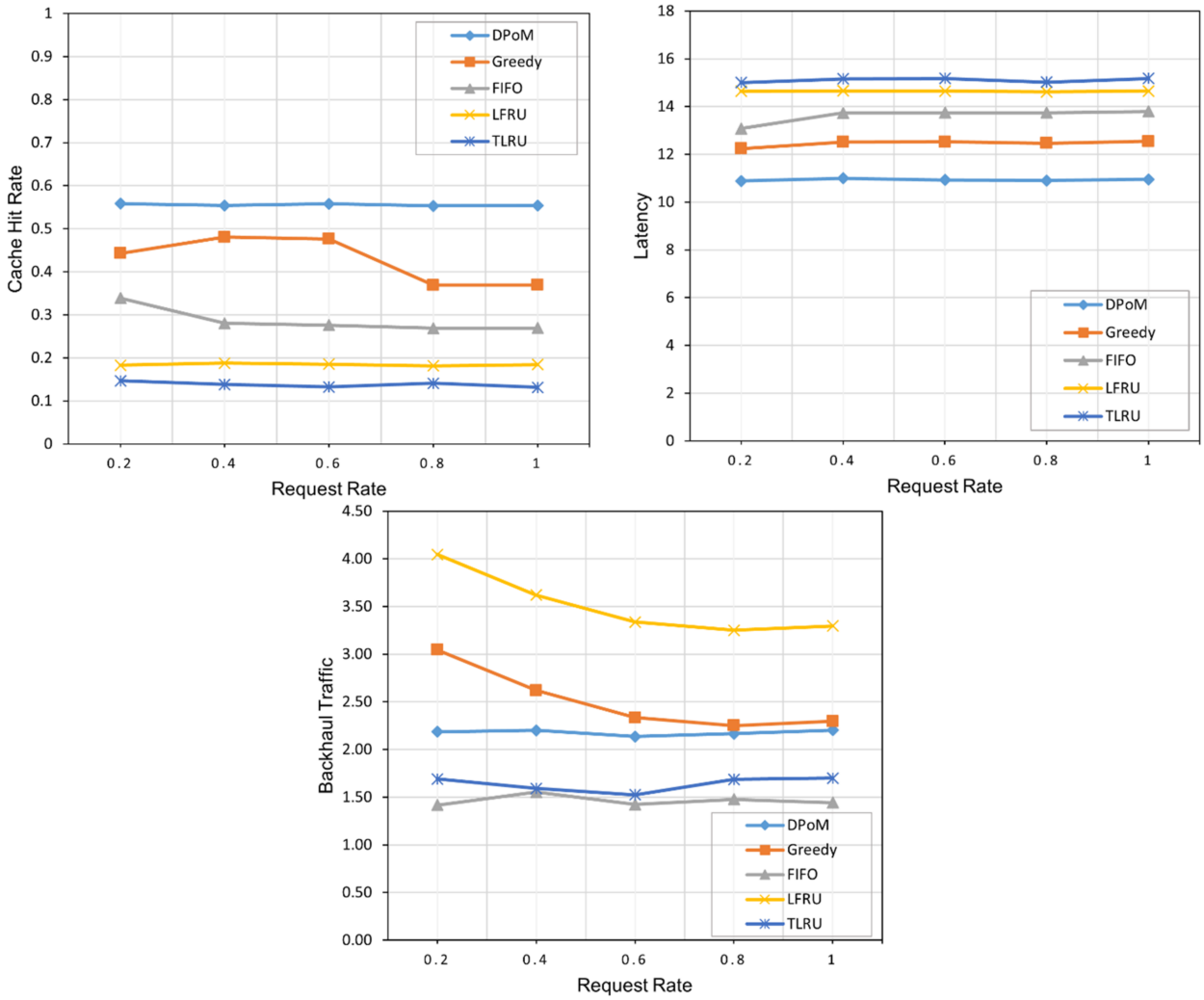

Figure 4. Performance of request rate variation on hit rate, latency, and backhaul traffic.

Latency performance on varying request rates shows a steady performance for all simulated caching schemes. The figure illustrates the good performance of the proposed scheme. The LFRU is outperformed by FIFO in this latency comparison. A reason for this is that the more requests the MAN servers receive, the better the chances of knowing and caching popular contents during cache replacement periods.

Increasing the request rates decreases the backhaul traffic of the edge caching network. More requests mean learning more about users' preferences. The learned user preferences are well exploited to inform the next cache decisions. Popularity tables of contents are updated when specific contents receive more requests than other files. In this way, caching files that are more likely to be requested reduces the traffic of transferring requested files from the ROC that have not been cached at the network edges.

5.4.3. Performance Evaluation under the Number of Users

In Figure 5, the performance of the caching schemes when the number of active users is varied is observed. The LFRU scheme dropped the initial good performance to average performance in this assessment. The best scheme for ensuring high hit rates with increasing number of active users is the proposed DPoM caching scheme. The FIFO and TLRU are the worst-performing schemes in this analysis.

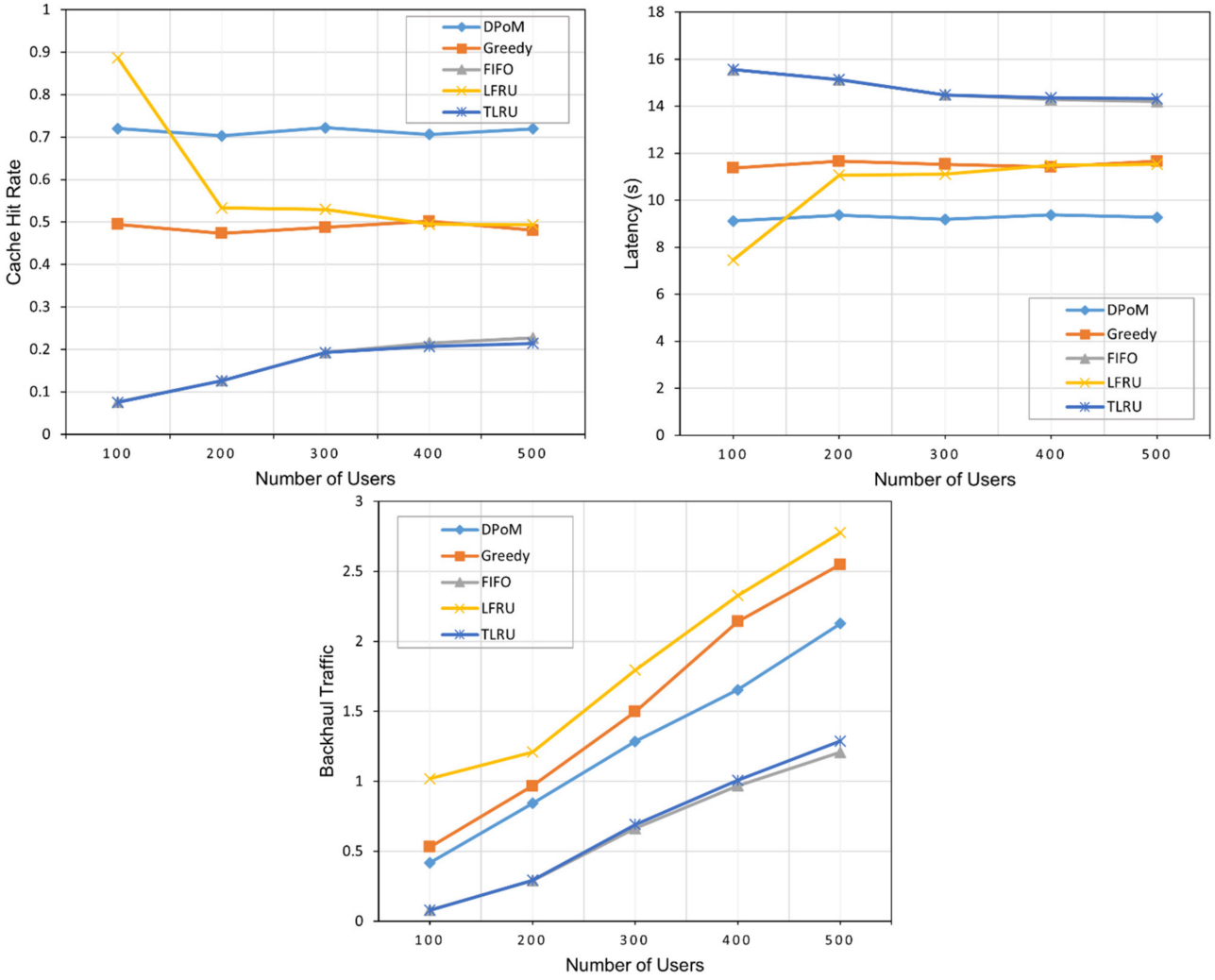

Figure 5. Performance of the number of users variation on hit rate, latency, and backhaul traffic.

The latency assessment realizes good performances from the proposed, LFRU and the Greedy schemes. The sharp decrease in latency, which is an advantage, is not realized here. In a cooperative device-to-device caching network, an increase in the number of active devices would have caused a decline in latency. Device-to-device caching is not considered in this work, as the underlying security features required to ensure fully functional device-to-device cooperative caching are not foundationally built. The issue of transmission power of the devices, bandwidth consumption among others further pushes the realization of this technology to a more distant future.

The third sub-figure of Figure 5 depicts that the backhaul traffic is directly proportional to the number of active users for all the caching schemes. The TLRU and FIFO are the best-performing schemes in this comparison. This can be attributed to the fact that these schemes transmit less data because of their caching policies, which do not involve content popularity considerations or fractional caching. Rather, entire files are cached and only replaced during cache replacements.

5.4.4. Performance Evaluation under Zipf Distribution of Content Files

The Zipfian distribution shows the closeness of the request probability of contents. A lower Zipf parameter indicates that only a few contents are likely to be requested many times while a higher Zipf parameter shows many content files having relatively same request probabilities, making all of the content files highly probable in the event of content requesting. Figure 6 presents the hit rate, latency, and backhaul traffic performances under

the varying Zipfian parameter. The cache hit rate plot shows a very close performance by the proposed, Greedy, and LFRU, but the best among them is the proposed. A lower Zipf parameter is good for the edge network because only a few contents are regularly requested. The most requested files can be comfortably cached to service users. On the other hand, if the parameter is high, the edge network needs to cache as many of the probable files to meet users' requests.

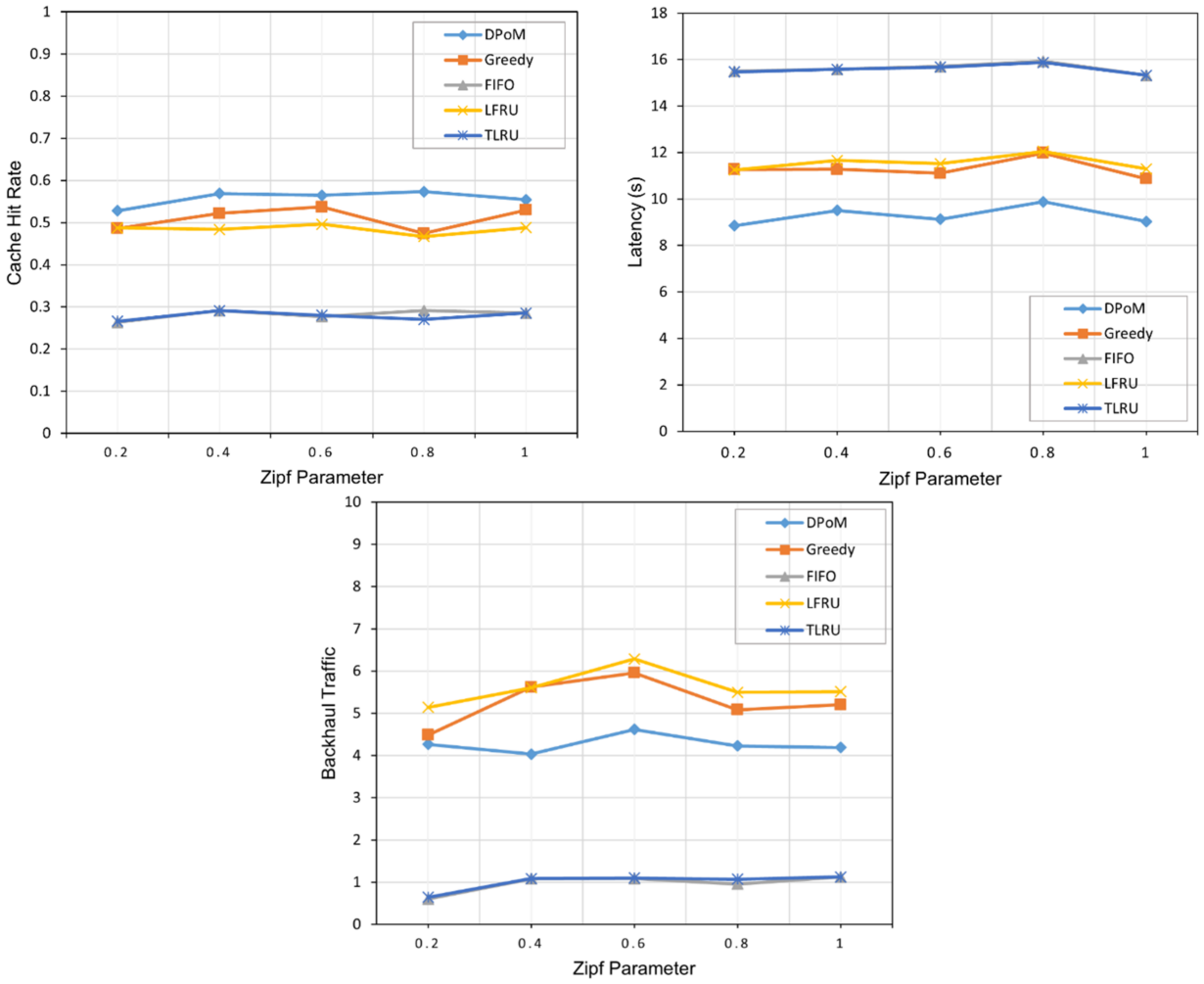

Figure 6. Performance of Zipf distribution on hit rate, latency, and backhaul traffic.

The latency and backhaul traffic records steady progressions of all the schemes with the increasing Zipf parameter. From the latency sub-figure of Figure 6, it can be deduced that DPoM outperforms the baseline schemes. However, the backhaul traffic performance show FIFO and the TLRU schemes outperforming the proposed, Greedy, and LFRU schemes.

5.4.5. Performance Evaluation under the Number of Files

Figure 7 illustrates cache hit rate, latency, and backhaul traffic performances when the number of cacheable contents is varied.

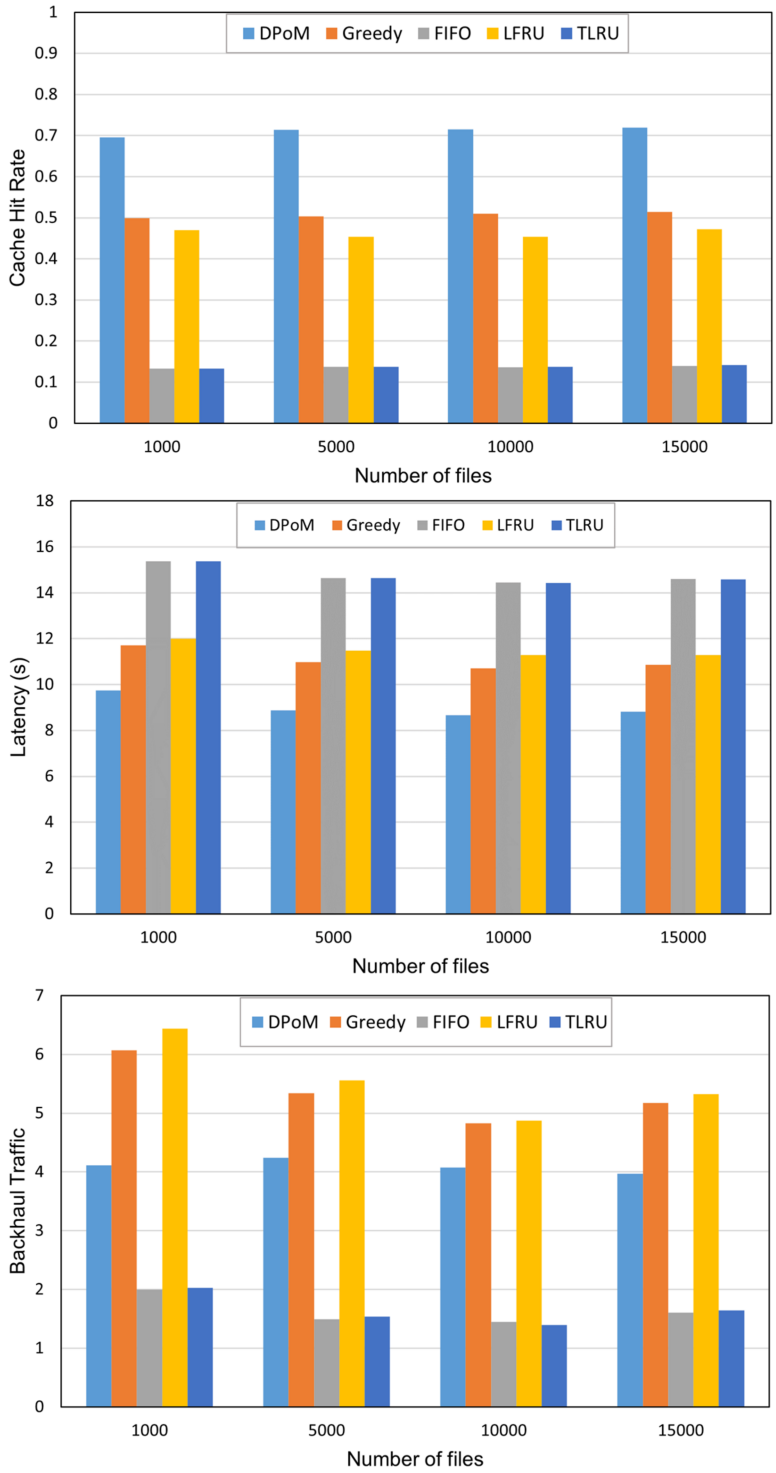

Figure 7. Performance of varying number of files on hit rate, latency, and backhaul traffic.

Similar to all hit rate comparisons, DPoM shows significant performance gains over the rest of the caching schemes. It utilizes the file splitting technique to store as many files as possible. An increase in the number of files does not cause any damaging impact on the proposed scheme.

The proposed scheme outperforms the baseline schemes in the latency comparison shown in the second sub-figure of Figure 7. The increase in the number of files had little effect on the caching schemes. It is observed that the increase in the number of files slowly

decreased the latency, which might not be thoroughly experienced during streaming since the change is not drastic.

The backhaul traffic comparison depicts the same observations as seen in previous comparisons. The proposed scheme is an average-performing scheme when backhaul traffic is considered. The amount of data transferred by DPoM is massive. This led to the recording of high backhaul traffic in the analyses.

6. Conclusions

In this paper, we proposed a part of media caching for edge caching networks aimed at increasing the number of contents cached at the network edge to achieve high cache hit rates and low latency video streaming services. To this end, we first defined the entities of our proposed caching scheme and formulated the edge caching problem as an Integer Linear Programming problem. Additionally, we proposed a file view-time threshold for each cached video to save backhaul and core network resources that is wasted when streaming sessions are abandoned leading to the discarding of the buffered content. We designed and employed heuristic algorithms to find and optimally place cacheable content in clustered cooperative MAN servers. Numerical results revealed the effectiveness of our proposed caching scheme over several state-of-the-art schemes, such as FIFO, Greedy, LFRU, and TLRU in achieving high hit rates, lower latencies, and average backhaul traffic. The reason behind the performance gains achieved by our method in comparison to existing considered baselines is caching only the first parts of the most popular contents, and delivering the remaining parts when the view-time threshold set on each requested file is reached.

As future work, we aim to extend the approach to caching new video applications and presentations intelligently at the network edge by exploiting the data characteristics of these new and immerging media. With the advancement of new video technologies and applications like 3D hologram, 3D and 360-degree video, and extended reality, the transmission of massive video data can cause bottlenecks for current and future networks. The purpose is to design and test caching and transmission schemes that can efficiently manage the data formats of new video services considering edge network heterogeneity and device compatibility for low latency retrieval. Federated learning approaches will be investigated to enhance the proposed approach for advanced functionality design.

Author Contributions: Conceptualization, E.O.-M. and S.K.S.T.; methodology, E.O.-M.; software, E.O.-M. and S.K.S.T.; validation, O.B. and E.A.-A.; formal analysis, O.B. and H.A.; investigation, E.O.-M. and C.L.; resources, S.K.S.T.; data curation, E.A.-A. and I.O.N.; writing—original draft preparation, E.O.-M., C.L. and I.O.N.; writing—review and editing, E.O.-M., O.B., C.L. and H.A.; visualization, E.O.-M. and S.K.S.T. All authors have read and agreed to the published version of the manuscript.

Funding: This article was funded in part by the National Natural Science Foundation of China under Grants 61871096.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank all contributors and Lab of Advanced Visual Communications and Computing-UESTC for supporting this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. *Cisco Visual Networking Index (VNI) Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper*; CISCO: San Jose, CA, USA, 2019.
2. Cisco. *2021 Global Networking Trends Report*; Cisco: San Jose, CA, USA, 2021.
3. Ge, C.; Wang, N.; Selinis, I.; Cahill, J.; Kavanagh, M.; Liolis, K.; Politis, C.; Nunes, J.; Evans, B.; Rahulan, Y.; et al. QoE-Assured Live Streaming via Satellite Backhaul in 5G Networks. *IEEE Trans. Broadcast.* **2019**, *65*, 381–391. [CrossRef]
4. Li, C.; Toni, L.; Zou, J.; Xiong, H.; Frossard, P. QoE-Driven Mobile Edge Caching Placement for Adaptive Video Streaming. *IEEE Trans. Multimed.* **2018**, *20*, 965–984. [CrossRef]

5. Liu, M.; Teng, Y.; Yu, F.R.; Leung, V.C.M.; Song, M. A Mobile Edge Computing (MEC)-Enabled Transcoding Framework for Blockchain-Based Video Streaming. *IEEE Wirel. Commun.* **2020**, *27*, 81–87. [CrossRef]
6. Mehrabi, A.; Siekkinen, M.; Yla-Jaaski, A. Edge computing assisted adaptive mobile video streaming. *IEEE Trans. Mob. Comput.* **2019**, *18*, 787–800. [CrossRef]
7. Yang, S.R.; Tseng, Y.J.; Huang, C.C.; Lin, W.C. Multi-Access Edge Computing Enhanced Video Streaming: Proof-of-Concept Implementation and Prediction/QoE Models. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1888–1902. [CrossRef]
8. Hoang, D.T.; Niyato, D.; Nguyen, D.N.; Dutkiewicz, E.; Wang, P.; Han, Z. A dynamic edge caching framework for mobile 5G networks. *IEEE Wirel. Commun.* **2018**, *25*, 95–103. [CrossRef]
9. Porambage, P.; Okwuibe, J.; Liyanage, M.; Ylianttila, M.; Taleb, T. Survey on Multi-Access Edge Computing for Internet of Things Realization. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2961–2991. [CrossRef]
10. Ndikumana, A.; Ullah, S.; LeAnh, T.; Tran, N.H.; Hong, C.S. Collaborative cache allocation and computation offloading in mobile edge computing. In Proceedings of the 19th Asia-Pacific Network Operations and Management Symposium: Managing a World of Things, APNOMS, Seoul, Korea, 27–29 September 2017. [CrossRef]
11. Tran, T.X.; Pompili, D. Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2019**, *18*, 1965–1978. [CrossRef]
12. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
13. Gul, F.; Mir, I.; Abualigah, L.; Sumari, P.; Forestiero, A. A consolidated review of path planning and optimization techniques: Technical perspectives and future directions. *Electronics* **2021**, *10*, 2250. [CrossRef]
14. Han, Y.; Wang, X.; Leung, V.C.M.; Niyato, D.; Yan, X.; Chen, X. Convergence of edge computing and deep learning: A comprehensive survey. *arXiv* **2019**, *22*, 869–904.
15. Bastug, E.; Bennis, M.; Debbah, M. Living on the edge: The role of proactive caching in 5G wireless networks. *IEEE Commun. Mag.* **2014**, *52*, 82–89. [CrossRef]
16. Rottenstreich, O.; Tapolcai, J. Lossy compression of packet classifiers. In Proceedings of the ANCS 2015—11th 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, Oakland, CA, USA, 7–8 May 2015; pp. 39–50. [CrossRef]
17. Kellerer, H.; Pferschy, U.; Pisinger, D. *Knapsack Problems*; Springer: Berlin, Germany, 2004. [CrossRef]
18. Plakia, M.; Tzamousis, E.; Asvestopoulou, T.; Pantermakis, G.; Filippakis, N.; Schulzrinne, H.; Kane-Esrig, Y.; Papadopouli, M. Should i Stay or Should i Go: Analysis of the Impact of Application QoS on User Engagement in YouTube. *ACM Trans. Model. Perform. Eval. Comput. Syst.* **2020**, *5*, 1–23. [CrossRef]
19. Gregori, M.; Gómez-Vilardebó, J.; Matamoros, J.; Gunduz, D. Wireless content caching for small cell and D2D networks. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 1222–1234. [CrossRef]
20. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. ETSI White Paper #11 Mobile Edge Computing—A key technology towards 5G. *ETSI White Pap. No. 11 Mob.* **2015**, *11*, 1–16.
21. Golrezaei, N.; Shanmugam, K.; Dimakis, A.G.; Molisch, A.F.; Caire, G. FemtoCaching: Wireless video content delivery through distributed caching helpers. In Proceedings of the Proceedings—IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012. [CrossRef]
22. Sukhmani, S.; Sadeghi, M.; Erol-Kantarci, M.; El Saddik, A. Edge Caching and Computing in 5G for Mobile AR/VR and Tactile Internet. *IEEE MultiMedia* **2019**, *26*, 21–30. [CrossRef]
23. Sonmez, C.; Ozgovde, A.; Ersoy, C. Performance evaluation of single-tier and two-tier cloudlet assisted applications. In Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 21–25 May 2017; pp. 302–307. [CrossRef]
24. Goian, H.S.; Al-Jarrah, O.Y.; Muhaidat, S.; Al-Hammadi, Y.; Yoo, P.; Dianati, M. Popularity-Based Video Caching Techniques for Cache-Enabled Networks: A Survey. *IEEE Access* **2019**, *7*, 27699–27719. [CrossRef]
25. Kumar, S.; Doddala, S.V.; Franklin, A.A.; Jin, J. RAN-aware adaptive video caching in multi-access edge computing networks. *J. Netw. Comput. Appl.* **2020**, *168*, 102737. [CrossRef]
26. Dehghan, M.; Massoulie, L.; Towsley, D.; Menasche, D.S.; Tay, Y.C. A Utility Optimization Approach to Network Cache Design. *IEEE/ACM Trans. Netw.* **2019**, *27*, 1013–1027. [CrossRef]
27. Ahlehagh, H.; Dey, S. Video-aware scheduling and caching in the radio access network. *IEEE/ACM Trans. Netw.* **2014**, *22*, 1444–1462. [CrossRef]
28. Shuja, J.; Bilal, K.; Alasmary, W.; Sinky, H.; Alanazi, E. Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey. *J. Netw. Comput. Appl.* **2021**, *181*, 103005. [CrossRef]
29. Wang, C.; Li, R.; Li, W.; Qiu, C.; Wang, X. SimEdgeIntel: A open-source simulation platform for resource management in edge intelligence. *J. Syst. Archit.* **2021**, *115*, 102016. [CrossRef]
30. Cheng, P.; Ma, C.; Ding, M.; Hu, Y.; Lin, Z.; Li, Y.; Vucetic, B. Localized Small Cell Caching: A Machine Learning Approach Based on Rating Data. *IEEE Trans. Commun.* **2019**, *67*, 1663–1676. [CrossRef]
31. Ugwuanyi, E.E.; Iqbal, M.; Dagiuklas, T. A novel predictive-collaborative-replacement (PCR) intelligent caching scheme for multi-access edge computing. *IEEE Access* **2021**, *9*, 37103–37115. [CrossRef]

32. Su, Z.; Xu, Q.; Hou, F.; Yang, Q.; Qi, Q. Edge Caching for Layered Video Contents in Mobile Social Networks. *IEEE Trans. Multimed.* **2017**, *19*, 2210–2221. [CrossRef]
33. Kim, Y.; Huh, E.N. EDCrammer: An efficient caching rate-control algorithm for streaming data on resource-limited edge nodes. *Appl. Sci.* **2019**, *9*, 2560. [CrossRef]
34. Tran, T.X.; Pandey, P.; Hajisami, A.; Pompili, D. Collaborative multi-bitrate video caching and processing in Mobile-Edge Computing networks. In Proceedings of the 2017 13th Annual Conference on Wireless On-Demand Network Systems and Services, WONS 2017—Proceedings, Jackson, WY, USA, 21–24 February 2017; pp. 165–172. [CrossRef]
35. Jiang, W.; Feng, G.; Qin, S.; Liu, Y. Multi-Agent Reinforcement Learning Based Cooperative Content Caching for Mobile Edge Networks. *IEEE Access* **2019**, *7*, 61856–61867. [CrossRef]
36. Zhang, S.; He, P.; Suto, K.; Yang, P.; Zhao, L.; Shen, X. Cooperative Edge Caching in User-Centric Clustered Mobile Networks. *IEEE Trans. Mob. Comput.* **2018**, *17*, 1791–1805. [CrossRef]
37. Paschos, G.S.; Iosifidis, G.; Tao, M.; Towsley, D.; Caireieee, G. The role of caching in future communication systems and networks. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1111–1125. [CrossRef]
38. Cheng, Y. Edge caching and computing in 5G for mobile augmented reality and haptic internet. *Comput. Commun.* **2020**, *158*, 24–31. [CrossRef]
39. Liu, M.; Yu, F.R.; Teng, Y.; Leung, V.C.M.; Song, M. Distributed Resource Allocation in Blockchain-Based Video Streaming Systems with Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 695–708. [CrossRef]
40. Zhan, W.; Luo, C.; Min, G.; Wang, C.; Zhu, Q.; Duan, H. Mobility-Aware Multi-User Offloading Optimization for Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3341–3356. [CrossRef]
41. Li, Q.; Lu, C.; Cao, B.; Zhang, Q. Caching resource management of mobile edge network based on Stackelberg game. *Digit. Commun. Netw.* **2019**, *5*, 18–23. [CrossRef]
42. Liu, Y.; Yu, F.R.; Li, X.; Ji, H.; Leung, V.C.M. Decentralized Resource Allocation for Video Transcoding and Delivery in Blockchain-Based System with Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2019**, *68*, 11169–11185. [CrossRef]
43. Forestiero, A.; Papuzzo, G. Agents-Based Algorithm for a Distributed Information System in Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 16548–16558. [CrossRef]
44. Kumar, S.; Vineeth, D.S.; Antony Franklin, A. Edge Assisted DASH Video Caching Mechanism for Multi-access Edge Computing. In Proceedings of the 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Indore, India, 16–19 December 2018; pp. 1–6. [CrossRef]
45. Gu, J.; Wang, W.; Huang, A.; Shan, H. Proactive storage at caching-enable base stations in cellular networks. In Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, London, UK, 8–11 September 2013. [CrossRef]
46. Baccour, E.; Erbad, A.; Bilal, K.; Mohamed, A.; Guizani, M. PCCP: Proactive Video Chunks Caching and Processing in edge networks. *Futur. Gener. Comput. Syst.* **2020**, *105*, 44–60. [CrossRef]
47. Tang, Y.; Rajendiran, D.P.; Moh, M. Cache Management for Cloud RAN and Multi-Access Edge Computing with Dynamic Input. In Proceedings of the 2019 International Conference on High Performance Computing & Simulation (HPCS), Dublin, Ireland, 15–19 July 2019; pp. 716–723. [CrossRef]
48. Wang, X.; Li, W. EdgeSim. Available online: https://github.com/XiaofeiTJU/EdgeSim.git (accessed on 10 June 2021).
49. Li, X.; Wang, X.; Wan, P.J.; Han, Z.; Leung, V.C.M. Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1768–1785. [CrossRef]
50. Banerjee, B.; Seetharam, A.; Tellambura, C. Greedy Caching: A Latency-aware Caching Strategy for Information-centric Networks.pdf. In Proceedings of the 2017 IFIP Networking Conference (IFIP Networking) and Workshops, Stockholm, Sweden, 12–16 June 2017. [CrossRef]
51. Bilal, M.; Kang, S.G. Time Aware Least Recent Used (TLRU) cache management policy in ICN. In Proceedings of the 16th International Conference on Advanced Communication Technology, Pyeongchang, Korea, 16–19 February 2014; pp. 528–532. [CrossRef]
52. Lee, D.; Choi, J.; Kim, J.H.; Noh, S.H.; Min, S.L.; Cho, Y.; Kim, C.S. LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE Trans. Comput.* **2001**, *50*, 1352–1361. [CrossRef]