EMS: Erasure-coded Multi-source Streaming for UHD Videos within Cloud Native 5G Networks

Lingjun Pu, Jianxin Shi, Xinjing Yuan, Xu Chen, Lei Jiao, Tian Zhang and Jingdong Xu.

Abstract—Ultra-High-Definition (UHD) videos have been getting increasing attention. However, existing video streaming solutions fail to deliver them due to the extremely high bandwidth requirement. The emerging cloud native 5G networks have opened up the possibility of enhancing UHD video quality by leveraging in-network video streaming. Unfortunately, the restricted storage and bandwidth of in-network servers could become the main bottleneck. To this end, we present EMS, a novel UHD video streaming framework, by integrating Erasure-coded storage with Multi-source Streaming. We respectively introduce a deadline-aware and a latency-sensitive metric to indicate the service quality of video servers and advocate a federated learning paradigm for the adaptive service quality update, including a reinforcement learning based multi-server selection (i.e., user local training) and a global service quality aggregation. To facilitate user local training without sacrificing streaming Quality-of-Experience (QoE), we cast the multi-server selection associated with the restriction on the average number of selected servers per video chunk into two kinds of Multi-Armed Bandit (MAB) models in terms of the proposed service quality metrics. We design lightweight Upper Confidence Bound (UCB) based algorithms with a theoretical performance guarantee. We implement a prototype of EMS, and extensive experiments confirm the superiority of the proposed algorithms.

Index Terms—Erasure-coded Storage, Multi-source Streaming, Cloud Native 5G Networks and Online Learning.

1 Introduction

Ultra-High-Definition (UHD) videos such as 8K/12K 2D videos, panoramic videos and volumetric videos have recently emerged and attracted great attention. In general, UHD videos have a large file size, since they are of high frame rate, high dynamic range and deep depth of field. Therefore, an extremely high downstream bandwidth is required to facilitate UHD video streaming [1], [2]. Although mobile network operators (MNOs) and Over-The-Top (OTT) content providers have made great efforts to extend the downstream bandwidth for video clients, such as building more base stations and utilizing multiple Content Delivery Networks (CDN) [3], [4], the achievable bandwidth (i.e., end-to-end throughput) is still inadequate, in terms of the data-driven and testbed-driven measurements in §2.1 (i.e., Fig. 3). The seemingly insatiable bandwidth demand of UHD videos motivates innovative video streaming solutions.

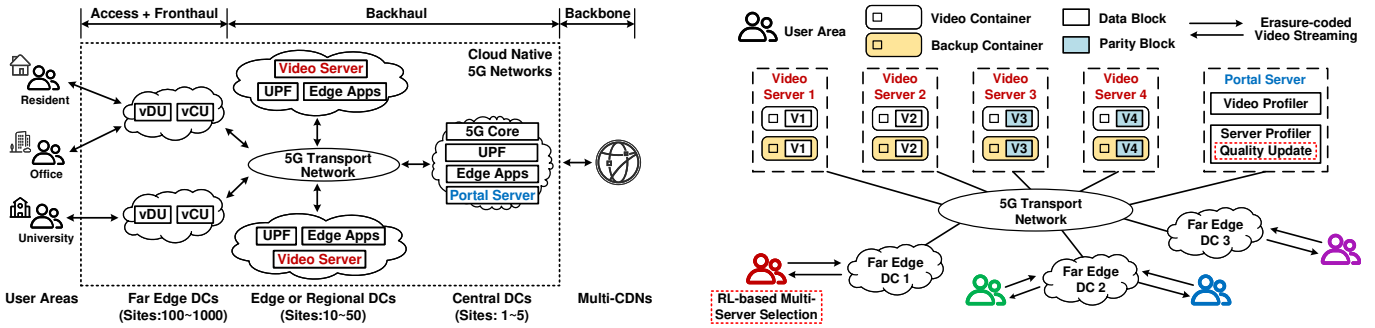
Cloud native 5G networks that embody containerization and micro-services have been receiving increasing attention from many MNOs such as Rakuten Mobile [5] and F5 Networks [6]. The reference architecture is shown in Fig. 1(a), where different scales of data centers (DCs) are hierarchically deployed to not only accomplish cellular tasks (e.g., UPF and 5G core) but also support various kinds of edge applications [7]. In this context, OTT content providers can cooperate with MNOs to deploy a number of video servers in the regional and/or central DCs, so as to achieve efficient UHD video streaming (e.g., low latency).

Lingjun Pu, Jianxin Shi, Xinjing Yuan, Tian Zhang and Jingdong Xu are with the College of Computer Science, Nankai University, Tianjin 300071, China. (e-mail: {pulj, shijx, yuanxj, zhangt, xujd}@nankai.edu.cn).
Xu Chen is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China. (e-mail: chenxu35@mail.sysu.edu.cn).
Lei Jiao is with the Department of Computer and Information Science, University of Oregon, Eugene 97403, USA. (e-mail: jiao@cs.uoregon.edu).

Although such an in-network video streaming solution can bring significant benefits such as making full use of advanced 5G cellular technologies [8] and getting rid of the bandwidth limitation of backbone network [9], it highly relies on the storage and bandwidth capacity of video servers due to the large file size of UHD videos, which however could not be guaranteed by those in-network DCs. This is because they are generally of small scales (e.g., the number of servers per regional DC is less than 30 as indicated by Huawei [10]), and accordingly have limited resources. What is worse, these limited resources will be primarily utilized for cellular tasks and the residual ones will be carved up by various edge applications. Therefore, the *restricted storage and bandwidth capacity of video servers (i.e., taking the form of containers) could be the main bottleneck of UHD video streaming*.

In order to alleviate the adverse effect of the restricted server bandwidth capacity, one popular idea is to reduce the transmission size while keeping the video quality. The representative solutions include viewpoint based streaming [1], [2] and super-resolution based streaming [11], [12], which however are still at an early research stage. For example, there are no perfect methods for accurate user viewpoint prediction, and super-resolution will produce uncertain and even intolerable processing delays. Another popular idea is to exploit multiple video servers to jointly transmit UHD videos (i.e., multi-source streaming [13], [14]). Existing solutions generally assume each video file is replicated to all the servers, and the bandwidth between client and server is accurately predicted. However, the former is unsuitable for the UHD video streaming within cloud native 5G networks due to the large video file size and the restricted server storage capacity, and the latter is very difficult to achieve in practice (e.g., Fig. 4).

As erasure-coded storage compared with replication based storage can provide space-optimal data redundancy [15]–[17], we advocate EMS, an Erasure-coded Multi-source Streaming framework for UHD videos within cloud native 5G networks. The



(a) Network scenario (each user has a cellular connection).

(b) Schematic diagram (a video server is located in a regional or central DC).

Fig. 1: The overview of EMS (UHD video V is coded into $D = 2$ data blocks and $P = 2$ parity blocks).

portal server, as shown in Fig. 1, will encode each video into D data blocks and P parity blocks (i.e., $D + P$ video blocks) and assign them to $D + P$ video servers. In other words, each video server will store only one video block. In practice, regional DCs will set up a video container to store the assigned video block and reserve several backup containers for elastic scaling and load balancing as a video server. The portal server will maintain the metadata of each video and its associated video servers. When a user requests a UHD video, it will query the portal server to obtain the metadata of the corresponding video servers (e.g., IP addresses), then select at least D servers from them and download their stored video blocks to recover the playable video.

Intuitively, integrating erasure-coded storage with multi-source streaming, EMS is highly appropriate for the video servers with the restricted storage and bandwidth capacity (i.e., storing and transmitting video blocks rather than the whole video). Besides, EMS is compatible with advanced streaming technologies such as adaptive streaming, viewpoint based streaming and super-resolution based streaming as discussed in §3.1. Despite its profound benefits, we require to address a critical problem: *how to optimally select a set of video servers for each user, in order to achieve efficient UHD video streaming*? The desirable multi-server selection algorithm should satisfy the following two requirements:

(1) **High user QoE**. The downstream bandwidth is a widely used metric to indicate the service quality of video servers for users. However, accurate bandwidth measurement at present is still a grand challenge [18], [19], and existing solutions such as bandwidth prediction or probing are inefficient or costly [20], [21]. Therefore, a novel metric associated with an efficient measurement method that can effectively represent the service quality of video servers for users is required. Otherwise, multi-source streaming could easily give rise to the cask effect [14].

(2) **Low system cost**. Selecting a large number of video servers can intuitively alleviate the cask effect of multi-source streaming. However, EMS in practice should restrict the number of selected servers to avoid excessive redundant data transmission that aggravates the traffic burden of cloud native 5G networks.

In this paper, we respectively introduce a deadline-aware metric and a latency-sensitive metric in terms of the historical user downloading time of video chunks to indicate the service quality of video servers, and advocate a Federated Learning (FL) paradigm to measure it (i.e., requirement (1)), which consists of a Reinforcement Learning (RL) based multi-server selection (i.e., user local training) and a global service quality update. Briefly, we adopt the concept of "user area" [22], [23], and consider the service quality of video servers for a user area can approximately represent that for the users belonging to it. In this context, the UHD video streaming of the users belonging to the same user area can be viewed as an asynchronous federated learning campaign. That is, when a user requests a UHD video, it will obtain the current service quality of video servers for its associated user area, locally train it during the video playback by invoking the RL based multi-server selection, and uploads the final trained result to the portal server for global update (i.e., global aggregation) at the end of video streaming. Note that, as this process continues over time, the service quality of video servers for any user area will adaptively capture its particular traffic pattern or distribution.

To capture the restriction on the number of selected servers per requested video, we introduce a long-term constraint to ensure the average number of selected servers per video chunk cannot exceed a predefined threshold (i.e., requirement (2)). Since mobile devices in general have limited computing capacity, we cast the RL based multi-server selection associated with the long-term constraint into two kinds of Multi-Armed Bandit (MAB) models in terms of the proposed two service quality metrics and respectively design two efficient Upper Confidence Bound (UCB) based algorithms with a theoretical performance guarantee and low complexity. In the end, we implement a prototype of EMS and conduct extensive testbed driven experiments for performance evaluation. The main contributions are summarized as follows:

- We present EMS, a novel erasure-coded multi-source streaming framework for UHD videos within cloud native 5G networks. We provide the specific design of its key components, propose new metrics to indicate the service quality of video servers, and advocate a FL paradigm including an RL based multi-server selection and a global aggregation for the adaptive service quality update (§3).
- We cast the RL based multi-server selection associated with the restriction on the average number of selected servers (i.e., a long-term constraint) into two kinds of MAB models in terms of the proposed two service quality metrics and correspondingly design two lightweight UCB based algorithms with sub-linear regret bounds (§4).
- Extensive testbed driven experiments confirm that (i) the practical performance of the proposed algorithms coincides with the theoretical analysis; (ii) the proposed algorithms achieve superior performance compared with the state-of-the-art multi-source streaming algorithms and UCB based algorithms in various system settings; (iii) EMS is a lightweight framework in terms of multi-server selection and erasure-coded chunk decoding (§5).

2 Motivation and Related Work

In this section, we provide the motivation of EMS in terms of dataset-driven and testbed-driven measurements and then outline the related work to highlight the novelty of EMS.

2.1 Downstream Bandwidth Measurements

Background. In general, UHD videos such as 8K/12K 2D, panoramic and volumetric videos have a large file size due to the high frame rate, high dynamic range and deep depth of field, and accordingly an extremely high downstream bandwidth is required to facilitate UHD video streaming. For example, according to Netflix, the recommended bandwidth for 4K (3840×2160) 2D videos is at least 25 Mbps [24], and therefore it would be 100 Mbps for 8K (7680×4320) 2D videos by simple multiplication. Besides, the required bandwidth for 12K panoramic videos should be over 400 Mbps [1]. Recently, MNOs and OTT content providers have made great efforts to extend the downstream bandwidth, such as building more base stations, upgrading optical transport networks and utilizing multi-CDN [3], [4]. Therefore, it is natural to ask *whether the current downstream bandwidth (i.e., end-to-end throughput) can well support UHD video streaming*?

Dataset. In order to answer the above question, we consider the following two video streaming datasets:

(i) *Puffer* [25], a video streaming dataset (i.e., 2022.01 – 2022.04) released by Stanford University. This dataset collects a set of time-series states from both the client and server side during the video streaming, and it contains a "delivery_rate" field to indicate the downstream bandwidth per video chunk.

(ii) *E2E-5G* [26], a 5G trace dataset collected from a major Irish mobile operator, which is generated from two mobility patterns (static and car) and two application patterns (video streaming and file download). This dataset contains a "DL_bitrate" field to indicate the downstream bandwidth per second. We only consider the data from the static pattern.

Testbed. Furthermore, we build a simple testbed to measure the downstream bandwidth on *Tencent Video*, a mainstream OTT platform in China. As shown in Fig. 2, we consider a laptop and a Huawei 5G CPE Pro2 with a plugged-in China Mobile 5G SIM card and connect them by a gigabit network cable to roughly represent a 5G device[1]. We select a list of video URLs, including TV dramas, movies, shows and animations, and conduct the bandwidth measurement by exploiting Selenium ChromeDriver [27] to automatically request and play them in the Chrome browser (i.e., we create a Python script that controls the ChromeDriver to load the selected video URL to the browser and click the video player button). Each video is downloaded as a series of HTTPS request/response interactions, and the Chrome browser (i.e., the built-in DevTools) will decrypt all the captured HTTPS request/response headers and archive relevant header fields. We exploit the content type header field (i.e., "video/MP2T" and "video/mp4") to filter out the records of video chunks (i.e., streaming logs). In order to obtain diverse and sufficient streaming logs, we consider four resident locations and two university sites in two megacities of China and run the testbed for a month (three times a day: Morning (9:00), Noon (13:00) and Night (20:00)).

1. Note that we do not directly exploit 5G smartphones in our measurement, since there is no perfect client-side monitoring tool for video streaming and moreover the client-side monitoring tool may negatively impact video player activities such as rendering (i.e., resource competition).

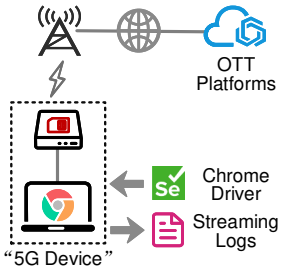


Fig. 2: Testbed overview.

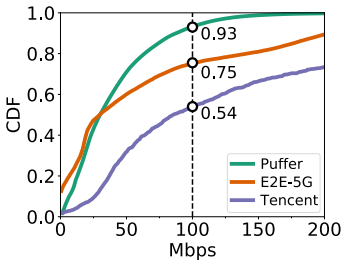


Fig. 3: CDF of bandwidth.

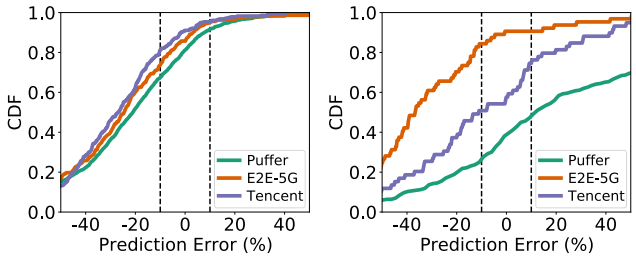
Measurement Result. We depict the CDF of the derived bandwidth samples as shown in Fig. 3. For clarity, we mark the point on the line whose x-axis value is 100. As indicated by them, we can derive that only 7%, 25% and 46% of total bandwidth samples of *Puffer*, *E2E-5G* and *Tencent Video* can satisfy the bandwidth requirement for 8K 2D videos (i.e., over 100 Mbps). Not to mention those UHD panoramic and volumetric videos. In other words, *the current achievable downstream bandwidth is still inadequate for UHD video streaming*, which motivates innovative video streaming solutions.

2.2 Related Work

Advanced streaming technology. Adaptive streaming such as DASH [28] is capable of coping with dynamic bandwidth fluctuations by adapting video quality in realtime. In adaptive streaming, each video is divided into a sequence of chunks with the same duration (e.g., several seconds), and each chunk is encoded with multiple discrete bitrates to accommodate various network conditions. The video client will sequentially request each chunk at an appropriate bitrate, in terms of a specific bitrate adaptation algorithm such as bandwidth based, buffer based and learning based algorithms [29]. However, adaptive streaming has little effect on the bandwidth increase.

Viewpoint or tile based streaming is a promising solution to save downstream bandwidth while keeping video quality, which is particularly tailored for panoramic and volumetric videos [1], [2]. Its basic idea is to transmit the tiles within the user viewpoint with a high bitrate and the other tiles with the lowest bitrate. However, this solution suffers from a fundamental limitation that user viewpoint is hard to predict accurately. Super-resolution based streaming is another promising solution [11], [12], whose basic idea is to recover high-resolution video frames from the transmitted lower resolution versions by using neural networks at the client side. However, it could produce uncertain and even intolerable processing delays, especially on mobile devices. Although EMS adopts a different design philosophy, it can be easily compatible with these promising yet immature solutions, as discussed in the fifth paragraph of §3.1.

Multi-source streaming. Multi-source streaming stems from the peer-to-peer (P2P) systems [30], and the P2P based solutions mainly follow the "one source per chunk" paradigm. For example, interplanetary file system (IPFS), the latest P2P system adopts a bandwidth probing solution (i.e., Bitswap [31]) to select the best peer for each requested content. However, EMS advocates a "multi-source per erasure-coded chunk" paradigm. In recent years, some researchers have considered bandwidth prediction based multi-source streaming [13], [32], in which they assume each video file is replicated to all the servers and the video



(a) Harmonic mean. (b) LSTM.

Fig. 4: Performance evaluation of bandwidth prediction algorithms.

client proportionally downloads each video chunk from all the servers in terms of the predicted bandwidth between the client and each server. Although they follow the "multi-source per chunk" paradigm, they are completely different from EMS in terms of the different storage and streaming mode. In addition, they will suffer from a fundamental limitation that the downstream bandwidth is hard to predict accurately [20], [21]. To support this argument, we evaluate the performance of the widely-used harmonic mean based [28] and LSTM based bandwidth prediction algorithm [21], in terms of the time-series bandwidth samples in the datasets and streaming logs given in §2.1. We adopt the prediction error as the metric which is calculated as (the predicted value − the actual value)/the actual value. The evaluation result is shown in Fig. 4, and we can find that as to each algorithm the percentage of "good" prediction whose error is within ±10% is roughly 18% (i.e., *Puffer*: 19%, 21.5%; *E2E-5G*: 21.5%, 7%; *Tencent Video*: 14.3%, 25.2% in Fig. 4).

Consider a simple scenario: two sources cooperatively transmit a 100Mb video chunk to client and the bandwidth between the client and each source is predicted to be 100Mbps. In this case, the expected transmission latency should be 0.5s under any bandwidth prediction based multi-source streaming algorithms (i.e., each source transmits 50Mb, 1/2 of video chunk). However, if the predicted downstream bandwidth of one source is underestimated (e.g., 120Mbps in fact), then the actual transmission latency is $\max\{50/120,\ 50/100\}$=0.5s while the optimal one should be 5/11=0.45s (i.e., the source with 120Mbps transmits 120/(120+100)=6/11 of video chunk and the source with 100Mbps transmits 100/(120+100)=5/11 of video chunk). On the other hand, if the predicted downstream bandwidth of one source is overestimated (e.g., 80Mbps in fact), then the actual transmission latency is 50/80=0.625s while the optimal one should be 5/9=0.56s (i.e., the source with 100Mbps transmits 5/9 of video chunk and the source with 80Mbps transmits 4/9 of video chunk). In both cases, the cask effect caused by the inaccurate bandwidth prediction results in 10% performance degradation. In addition, a large-scale and real-world measurement indicates that 23% of the downloads have worse performance after being upgraded to multi-source content delivery due to the cask effect [14]. As such, the multi-source streaming with inaccurate bandwidth prediction is suboptimal and its latency is determined by the slowest source.

Erasure-coded storage system. Erasure-coded storage has been extensively discussed in distributed storage systems as it can provide space-optimal data redundancy. Briefly, erasure coding works by splitting a file into multiple fragments (data blocks) and then creating additional fragments (parity blocks) that can be used for file recovery. The data and parity fragments are stored across multiple disks to protect against data loss in case a disk fails. If such an event occurs, the file can be rebuilt by using the available data and parity fragments. For example, a storage system could exploit a 5+2 encoding configuration, in which each file is splitted into five data fragments and then adds two parity fragments. The configuration can tolerate up to two disk failures, no matter whether the disks contain data or parity fragments. In other words, the file can be rebuilt by five data or parity fragments.

In recent years, some researchers focus on how to accurately quantify the access latency for erasure-coded storage systems, and they mainly exploit the queuing theory to derive and analyze the latency bounds [16], [17], [33]. For example, the authors in [16] consider video streaming over an erasure-coded cloud system and analyze the mean stall duration and the stall duration tail probability with complicated probabilistic models. Other researchers attempt to design efficient caching schemes to achieve a low access latency in erasure-coded storage systems. For example, the authors in [15] design an online erasure coding scheme on the cached data to achieve load balancing and latency reduction. There are also some edge caching studies taking erasure-coded storage into account and mainly discuss the content placement problem [34]–[37]. Nevertheless, none of the existing work like EMS pays attention to the multi-server selection problem in the context of integrating erasure-coded storage with multi-source streaming.

Multi-Armed Bandit (MAB) for video streaming. There are some recent studies that exploit multi-armed bandit to model bitrate selection [20], [38] or server selection [39], [40] for video streaming. However, they only require to select one arm, which is different from EMS. The most related work is [14], which attempts to select a fixed number of workers by proposing a variant of ϵ-greedy algorithm. However, the considered service quality metric is different from EMS (i.e., different objective functions). In addition, it does not involve a long-term constraint on the average number of selected servers. Moreover, its proposed algorithm does not provide a theoretical performance guarantee.

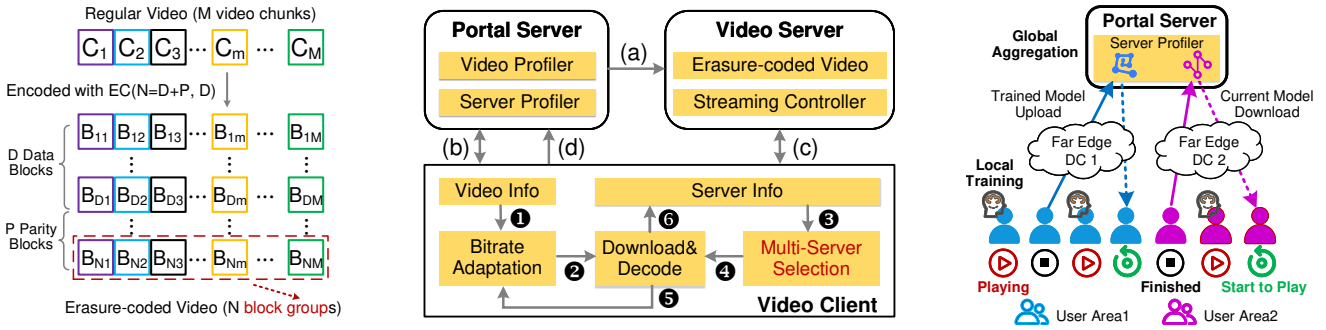
TABLE 1: Summary of related works on multi-source streaming.

Existing Works	Storage Mode	Server Selection Constraint	Algorithmic Bound
[30], [31]	P2P based	No	No
[13], [32]	Replication	No	No
[16]	Erasure-coding	No	Yes
[14]	P2P based	Fixed number	No
Ours	Erasure-coding	Long-term	Yes

Summary. We highlight the characteristics of the highly related multi-source streaming works in Table 1. To the best of our knowledge, EMS is the first work that integrates erasure-coded storage with multi-source streaming, concentrates on the multi-server selection problem with novel service quality metrics and a long-term constraint, and proposes efficient algorithms with performance guarantee and low complexity.

3 System Design

In this section, we provide the specific design of EMS including the basic components and interactions, then formulate the critical multi-server selection problem with a constrained reinforcement learning model.



(a) Illustration of erasure-coded video. (b) Basic components and interactions. (c) Federated learning based service quality.

Fig. 5: System design of EMS.

TABLE 2: Main notations.

Parameters	Descriptions
m	the index of video chunk that also indicates the round of multi-server selection during a video streaming
D	the number of data blocks per video chunk that also indicates the required number of video blocks to decode and recover each video chunk
P	the number of parity blocks per video chunk
t	the round of global aggregation at the portal server
$\theta_k^u(m)$	the local deadline-aware indicator
$\Theta_k^u(t)$	the global deadline-aware indicator
$\omega_k^u(m)$	the local latency-sensitive indicator
$\Omega_k^u(t)$	the global latency-sensitive indicator
$\mathcal{K}(t)$	the associated servers of the requested video at round t that also indicates the video servers that can be selected
H	the predefined threshold to limit the average number of selected servers per video chunk
Variables	**Descriptions**
$\mathcal{S}(m) \subseteq \mathcal{K}(t)$	the set of selected video servers for video chunk C_m

3.1 Basic Components and Interactions

As shown in Fig. 1(a), EMS consists of a portal server located in a central DC and a set $\mathcal{K} = \{1, 2, \ldots, K\}$ of video servers located in the regional DCs. Due to the resource limitation, we consider EMS only serves popular UHD videos which can be easily identified by OTT content providers in practice. In the following, we will introduce the basic system components and detail their functionalities along with the interactions occurring in the erasure-coded video deployment and streaming. The main notations in EMS are given in Table 2.

Erasure-coded video deployment. When the portal server receives a popular UHD video from OTT content providers, it will encode it to be an erasure-coded video, and then deploy it to several video servers. Specifically, as shown in Fig. 5(a), each video is generally divided into multiple chunks with the same duration (e.g., several seconds) denoted by $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$, and each chunk C_m is erasure-coded into D data blocks and P parity blocks (i.e., $D+P$ video blocks) denoted by $\{B_{1m}, B_{2m}, \ldots, B_{Nm}\}$ where $N = D+P$. In this context, we define a "block group" as a set of M video blocks satisfying no video blocks comes from the same video chunk. For example, we can exploit the set $G_n = \{B_{n1}, B_{n2}, \ldots, B_{nM}\}, \forall n \in \{1, 2, \ldots, N\}$ to indicate a block group. Naturally, the erasure-coded video is made of N block groups which are of the same file size.

In order to fully reap the benefit of multi-source streaming, we consider that the portal server will assign N block groups of an erasure-coded video to N different video servers. For example, it can randomly select N servers from the total K video servers[2]. In other words, each video server will store only one block group for each video, which does not consume too much storage resource and accordingly is highly suitable for the resource-restricted video servers within cloud native 5G networks. In practice, the regional DCs will set up a video container to store the assigned block group and reserve several backup containers for elastic scaling and load balancing (i.e., *erasure-coded video* component and **Step (a)** in Fig. 5(b)). That is, each video server in EMS is made of several containers. Besides, the portal server will create a *video profiler* component to keep the metadata of each UHD video (e.g., media presentation description (MPD) file) along with the identification of its associated video servers (e.g., IP addresses).

As to the storage redundancy of erasure-coded video compared with the original video, if we exploit $|C_m|$ to denote the file size of chunk C_m, then the size of its data block and that of its parity block are both $|C_m|/D$ due to the principle of erasure coding. In this context, the size of each block group G_n is $\sum_{m=1}^{M} |C_m|/D$ (i.e., $1/D$ of the original video), and the storage redundancy of each erasure-coded chunk is P/D. As such, the storage redundancy of erasure-coded video is also P/D, in terms of the following expression:

$$\frac{\sum_{m=1}^{M}(1+P/D)|C_m| - \sum_{m=1}^{M}|C_m|}{\sum_{m=1}^{M}|C_m|} = P/D,$$

where $(1+P/D)|C_m|$ represents the file size of erasure-coded chunk for each chunk C_m. If we exploit a 5+2 encoding configuration, then the storage redundancy is 0.4. Note that since we only consider popular UHD videos in EMS (i.e., the number of deployed videos is limited), we believe this erasure-coded video deployment is acceptable in practice.

Note that we can easily extend the scope of the block group to adapt to the UHD video with multiple discrete bitrates (e.g., R bitrate versions) so as to be compatible with the adaptive streaming. Briefly, we will introduce a "super block group" as a set of R block groups satisfying no block groups from the same bitrate version and assign N super block groups to N video servers. Similarly, we can also extend the scope of the block group to facilitate viewpoint based streaming (i.e., erasure-coding each tile rather than each chunk). Besides, the erasure-coded video has

2. How to select the optimal servers for the erasure-coded video deployment (i.e., content placement problem) is not the main focus of this paper.

no impact on the super-resolution based streaming. In other words, EMS is compatible with it with no modifications.

Erasure-coded video streaming. When a client starts a UHD video streaming, the portal server will receive the client request and check if the *video profiler* component has the metadata of the requested video. If not, the UHD video streaming will be provided by the general multi-CDN. If yes, the portal server will forward the corresponding video metadata (e.g., MPD file), the identification of the associated video servers (e.g., IP addresses), and the service quality of the associated video servers for that user maintained by the *server profiler* component to the client (i.e., **Step (b)**). When the client receives the information of the requested video and that of its associated video servers, it will respectively store them in the *video info* and *server info* component and then start the following erasure-coded video streaming.

In terms of the received MPD file, the client will invoke any bitrate adaptation algorithm (e.g., [21], [28]) to select an appropriate bitrate for the next video chunk (i.e., **Step ❶** and *bitrate adaptation* component). Then, the chunk index and the selected bitrate will be forwarded to the *download and decode* component (i.e., **Step ❷**). In the meanwhile, The *multi-server selection* component is invoked to obtain the received service quality of video servers stored in the *server info* component to make multi-server selection (i.e., **Step ❸**), and then passes the selected servers to the *download and decode* component (i.e., **Step ❹**). Next, the *download and decode* component will download the stored video blocks from those selected servers (i.e., **Step (c)**). The *streaming controller* component in each selected server will allocate the corresponding video container or one of its backup container to that client in terms of some predefined rules (e.g., round-robin or load balancing). When the *download and decode* component has successfully downloaded D video blocks, it will decode them to recover the playable video chunk, approximately calculate the achievable downstream bandwidth as the file size of the recovered video chunk divided by the download time of the D-th downloaded video block, and then send it to the *bitrate adaptation* component to facilitate subsequent bitrate adaptation (i.e., **Step ❺**). Note that the **Step ❶** – **Step ❺** will operate recurrently until the video streaming finishes.

3.2 Problem Formulation

In terms of the above discussions, we can easily find that the *multi-server selection* component (the core of EMS) requires to address a critical problem: *how to optimally select a set of video servers so as to achieve efficient UHD video streaming*? The desirable multi-server selection algorithm should satisfy the following two requirements:

(1) **High user QoE**. It is not easy to accurately indicate the service quality of video servers. Although the downstream bandwidth is a widely used metric, accurate measurement is still challenging [18], [19], which is also validated by our measurements in Fig. 4. Multi-server selection with inaccurate bandwidth prediction could easily give rise to the cask effect [14].

(2) **Low system cost**. Selecting a large number of video servers can effectively alleviate the cask effect of multi-source streaming. However, EMS in practice should restrict the number of selected servers to avoid excessive redundant data transmission.

Our idea. As to requirement (1), we will respectively introduce a deadline-aware metric and a latency-sensitive metric to indicate the service quality of video servers for users, since many studies have pointed out that users may quickly abandon a requested video if the number of rebuffering is large or the duration of rebuffering is long [20], [21], [28]. In order to efficiently measure the proposed metrics, we exploit the concept "user area"[3] which refers to a geographical location whose mobile traffic exhibits a particular pattern or distribution [22], [23], and further consider the service quality of video servers for a user area can approximately represent that for the users belonging to it. In this context, we advocate a Federated Learning (FL) paradigm for the adaptively update of service quality of video servers for each user area, which consists of a Reinforcement Learning (RL) based multi-server selection (i.e., local training at the client side) and a global service quality aggregation at the portal server side as shown in Fig. 5(c). As to requirement (2), we will introduce a long-term constraint to ensure the average number of selected servers per video chunk cannot exceed a predefined threshold, which is embedded into the RL based multi-server selection. Note that such a soft constraint can extend the flexibility of multi-server selection in the context of dynamic network conditions. The specific design is given as follows.

Deadline-aware metric. We consider this metric to capture the scenario that users dislike a large number of rebuffering during the video streaming. Motivated by many OTT platforms such as Youtube and Netflix have given the recommended bandwidth for different types of videos, we introduce the "unit download deadline" of each video type v (e.g., 8K/12K and 2D/3D) as $\tau_{max}(v) = 1/W_v$, where W_v refers to the lowest required bandwidth (in Mbps). For simplicity, we only consider one video type and omit the index v in the following. Then, we can derive the download deadline of any a video block of chunk C_m as $T_{max}(m) = F_m\tau_{max}$, where F_m refers to the file size of the video block[4] of chunk m. To proceed, we introduce $s_k(m)$ to indicate whether the latency[5] $l_k(m)$ of downloading a video block of chunk C_m from server k is smaller than the deadline. Intuitively, $s_k(m) = 1$ if $l_k(m) \leq T_{max}(m)$ and 0 otherwise. In this context, we define the deadline-aware metric as *the probability of successfully transmitting a video block before its corresponding deadline*, and exploit $\{\theta_k^u(m), \Theta_k^u(t)\}$ to indicate the local and global service quality of server k for user area u. Here, m refers to the index of video chunk that also indicates the round of multi-server selection during a video streaming, and t refers to the round of global aggregation kept by the portal server. The specific $\theta_k^u(m)$ is given in the expression (4).

Latency-sensitive metric. We consider this metric to capture the scenario that users also dislike a long duration of rebuffering during the video streaming. Then, we introduce the "normalized truncated latency" of downloading any a video block B_{nm} from video chunk C_m as $l'_k(m) = \min\{l_k(m), T_{max}(m)\}/T_{max}(m)$, where $l_k(m)$ and $T_{max}(m)$ share the same meanings as those mentioned above. To maintain consistency with the deadline-aware metric, we define the latency-sensitive metric as *the average gap between the chunk download latency and the its corresponding deadline*, and exploit

3. Note that "user area" has been widely considered in the research of edge computing and edge caching, where multiple edge servers jointly satisfy the computation tasks and content requests from any user area. In practice, user area can be indicated by a set of base stations that are in proximity to each other [22], [23].

4. The data blocks and parity blocks are of the same file size in the context of erasure-coding. The file size of video block of each video chunk can be viewed as one kind of video metadata stored by the *video profiler* component.

5. This value can be derived from the *download and decode* component.

$\{\omega_k^u(m), \Omega_k^u(t)\}$ to indicate the local and global service quality of server k for user area u. In practice, $\omega_k^u(m)$ is updated by $1-l'(m)$. Intuitively, the local service quality $\theta_k^u(m)$ and $\omega_k^u(m)$ are both within $[0,1]$ and a larger value indicates a higher service quality of server k for user area u.

FL based adaptive update of service quality. We regard the UHD video streaming of the users belonging to the same user area as an asynchronous FL task. Taking the deadline-aware metric as an example, when a user $i \in u$ requests an UHD video, it will obtain the current global service quality indicator $\Theta_k^u(t)$ from the *server profiler* component, train the local service quality indicator $\theta_k^u(m), \forall m \in \{1,2,\ldots,M\}$ with $\theta_k^u(0) = \Theta_k^u(t)$ during the video playback (i.e., **Step ❻** in Fig. 5(b)), and upload the trained indicator $\theta_k^u(M)$ to the *server profiler* component for global aggregation (i.e., **Step (d)** in Fig. 5(b)). Note that the specific aggregation method is not the main focus of this paper. In practice, we can simply update the global indicator by $\Theta_k^u(t+1) = \big[\Theta_k^u(t) * t + \theta_k^u(M)\big]/(t+1)$ or adopt advanced methods from asynchronous FL studies (e.g., [41]).

We believe this asynchronous procedure is ease to implement and does not affect user video streaming in practice. We can also conduct the global aggregation per video chunk from multiple video clients in a synchronous manner and then distribute the updated parameters back to those clients. However, it could introduce additional synchronization time and transmission overhead for training, which inevitably damages the performance of user video streaming.

RL based multi-server selection. To facilitate user local training without sacrificing the QoE of UHD video streaming, we consider multi-server selection is invoked per video chunk and model it with reinforcement learning. To proceed, we will omit the user area index u for clarity and exploit the server set $\mathcal{K}(t)$ to indicate the video servers that can be selected (i.e., the associated servers of the requested video) at round t. Then, we can define the state, action and reward as follows.

- **State**: the local service quality indicator $\theta_k(m)$ or $\omega_k(m), \forall m \in \{1,2,\ldots,M\}$ of each server $k \in \mathcal{K}(t)$.
- **Action**: a subset $\mathcal{S}(m) \subseteq \mathcal{K}(t)$ of video servers that satisfies the following long-term constraint for the restriction on the number of selected servers:

$$\frac{1}{M}\sum\nolimits_{m=1}^{M} |\mathcal{S}(m)| \leq H, \tag{1}$$

  where H refers to a predefined server selection threshold.
- **Reward**: we adopt $r\big(\mathcal{S}(m)\big)$ to uniformly indicate the reward when selecting the server subset $\mathcal{S}(m)$. As to the deadline-aware metric, it should be

$$\begin{aligned} &r\big(\mathcal{S}(m)\big) = \sum\nolimits_{x=D}^{|\mathcal{S}(m)|} \mathbb{P}\big(|X(m)| = x\big), \\ &\mathbb{P}\big(|X(m)| = x\big) \triangleq \sum_{X(m)\subseteq\mathcal{S}(m), |X(m)|=x} \mathbb{P}\big(X(m)\big), \\ &\mathbb{P}\big(X(m)\big) \triangleq \prod_{k\in X(m)} \theta_k(m) \prod_{k\in\mathcal{S}(m)\setminus X(m)} (1-\theta_k(m)). \end{aligned} \tag{2}$$

  Here, $X(m)$ refers to the "successful" subset of $\mathcal{S}(m)$ and $|X(m)| = x$ refers to the event that the size of the "successful" subset is x. In this context, $r\big(\mathcal{S}(m)\big)$ can be interpreted as the probability of successfully downloading at least D video blocks (i.e., the required number of video blocks to decode and recover the video chunk) from $\mathcal{S}(m)$ within the deadline.

  As to the latency-sensitive metric, it should be

$$\begin{aligned} &r\big(\mathcal{S}(m)\big) = \min\nolimits_{k\in\mathcal{S}'(m)} \omega_k(m), \\ &\mathcal{S}'(m) \triangleq \big\{\mathcal{S}'(m) : |\mathcal{S}'(m)| = D\big\} \subseteq \mathcal{S}(m). \end{aligned} \tag{3}$$

  Here, $|x|$ refers to the size of set x and $r\big(\mathcal{S}(m)\big)$ can be interpreted as the time consumption of successfully downloading D video blocks from $\mathcal{S}(m)$, which is obviously determined by the D-th fastest server (i.e., the D-th largest average gap between latency and deadline).
- **Update**: take the local service quality indicator $\theta_k(m)$ as an example[6] and it will be updated at runtime as follows:

$$\theta_k(m) = \begin{cases} \frac{\theta_k(m-1)n_k(m-1)+s_k(m)}{n_k(m-1)+1} & \text{if } k \in \mathcal{S}(m), \\ \theta_k(m-1) & \text{else}, \end{cases} \tag{4}$$

  where $n_k(m-1)$ refers to the times the server k is selected at the end of round $m-1$, which is updated as follows:

$$n_k(m) = \begin{cases} n_k(m-1)+1 & \text{if } k \in \mathcal{S}(m), \\ n_k(m-1) & \text{else}. \end{cases} \tag{5}$$

To sum up, we can uniformly formulate a constrained RL based multi-server selection problem as follows:

$$\max_{\{\mathcal{S}(m), m\geq 1\}} \sum\nolimits_{m=1}^{M} r\big(\mathcal{S}(m)\big), \text{ subject to } (1).$$

Note that it is non-trivial to solve this problem, mainly due to the following issues:

(i) The general RL models do not involve a long-term constraint (i.e., constraint (1)), and accordingly we need to provide a specific technique to cope with it. In addition, the unique feature of the above two reward expressions (e.g., nonlinearity) further complicates the efficient algorithm design;

(ii) The proposed multi-server selection algorithm should be lightweight (e.g., at the timescale of milliseconds). Otherwise, it will offset the benefit of multi-source streaming.

4 Multi-server Selection Algorithm

In this section, we cast the above RL problem into two different Multi-Armed Bandit (MAB) models in terms of the proposed two service quality metrics, and respectively design two lightweight Upper Confidence Bound (UCB) based algorithms with a theoretical performance guarantee. We first tackle the more straightforward latency-sensitive multi-server selection and then delve into the deadline-aware multi-server selection.

4.1 Latency-sensitive multi-server selection

Problem reformulation. In the MAB model, we regard each server $k \in \mathcal{K}$ as an arm, and the selected subset $\mathcal{S}(m)$ is referred to as a super arm. To begin with, we adopt the widely used *regret* metric to reformulate our problem. Briefly, we denote by $\omega_k^* = \mathbb{E}[\omega_k(m)]$ the expectation of local service quality of arm k. In this case, the expected optimal super arm is $\mathcal{S}^* = \underset{\mathcal{S}\subseteq\mathcal{K}(t), |\mathcal{S}|=D}{\arg\max} \big\{\min_{k\in\mathcal{S}} \omega_k^*\big\}$. Then, we can define the cumulative regret of any given algorithm as

$$\Delta_{ALG} = Mr(\mathcal{S}^*) - \mathbb{E}\Big[\sum\nolimits_{m=1}^{M} r\big(\mathcal{S}(m)\big)\Big], \tag{6}$$

6. Note that for the local service quality indicator $\omega_k(m)$ the only difference in the expression (4) is to substitute $s_k(m)$ with $1-l'_k(m)$.

Algorithm 1: L-EMS

Input: $\Omega_k(t)$, $\mathcal{K}(t)$, D
Output: $\mathcal{S}^*(m)$, $m \in \{1, 2, \ldots, M\}$
▷ *Initial model download from the portal server*
$\omega_k(0) = \Omega_k(t), n_k(0) = 1, k \in \mathcal{K}(t)$;
▷ *RL based local training at the client side*
For each video chunk $m \in \{1, 2, \ldots, M\}$ **do**
  Calculate $\widetilde{\omega}_k(m)$ by the expression (8);
  Sort $\{\widetilde{\omega}_k(m), k \in \mathcal{K}(t)\}$ in the descending order;
  Select the top D arms from the sorted order as $\mathcal{S}^*(m)$;
  Update $\omega_k(m)$ by the expression (4) (i.e., substituting $\theta_k(m)$ and $s_k(m)$ with $\omega_k(m)$ and $1 - l'_k(m)$);
▷ *Asynchronous global aggregation at the portal server side*
Update $\Omega_k(t+1)$ by $\omega_k(M)$ such as
  $\Omega_k(t+1) = \big[\Omega_k(t) * t + \omega_k(M)\big]/(t+1)$.

where $r(\mathcal{S}^*) = \min_{k \in \mathcal{S}^*} \omega_k^*$ refers to the expected reward achieved by the super arm $\mathcal{S}^*$. According to the expression (3), we can observe that there is no benefit to play more than D arms in $\mathcal{S}(m)$. This observation motivates us to reformulate the constraint (1) as $|\mathcal{S}(m)| = D, \forall m \in \{1, 2, \ldots, M\}$. In this context, we can derive the reformulated problem for latency-sensitive multi-server selection as follows:

$$\min_{\{\mathcal{S}(m), m \geq 1\}} \Delta_{ALG}, \text{ subject to } |\mathcal{S}(m)| = D. \tag{7}$$

Algorithm design. According to the above formulation and the expression (3), we can observe the feedback for each arm k independently rather than for each super arm $\mathcal{S}$ as a whole. In this context, we will design a UCB based algorithm for such a classic MAB problem. Specifically, we denote by $\widetilde{\omega}_k(m)$ the UCB estimate of the service quality of arm k at the beginning of the local training round m which is given by

$$\widetilde{\omega}_k(m) \triangleq \omega_k(m-1) + \sqrt{(D+1)\ln m / n_k(m-1)}, \tag{8}$$

where $\omega_k(m-1)$ and $\sqrt{(D+1)\ln m / n_k(m-1)}$ respectively correspond to exploitation and exploration, and $n_k(m)$ is updated by the expression (5). Then, our solution for the problem in (7) is to select a super arm per round satisfying

$$\mathcal{S}^*(m) = \underset{\mathcal{S} \subseteq \mathcal{K}(t), |\mathcal{S}| = D}{\arg\max} \big\{ \min_{k \in \mathcal{S}} \widetilde{\omega}_k(m) \big\}. \tag{9}$$

Note that we can easily derive $\mathcal{S}^*(m)$ by sorting the arms in the descending order of their UCB estimates and selecting the top D arms. The detailed algorithm L-EMS are given in Alg. 1.

Performance analysis. According to Alg. 1, we can know that the complexity of latency-sensitive multi-server selection is lightweight (i.e., dominated by the sorting). In addition, the expected cumulative regret can be upper bounded by

$$\Delta_{ALG} \leq \delta \Big[2K_t + \frac{\pi^2}{3} D K_t + \frac{4D^2(D+1)K_t \ln M}{\delta'^2} \Big],$$

where both δ and δ' are constant values during user local training, and K_t refers to the size of $\mathcal{K}(t)$. The detailed proof is provided in our online technical report [42]. Clearly, the cumulative regret grows as $\mathcal{O}(\ln M)$, which is strictly in a logarithmical way in the number of local training rounds (i.e., the number of video chunks). In addition, according to the discussions in the third paragraph of §3.1, K_t by definition equals to the number of video blocks (i.e., $D + P$). Therefore, it seems that we should set a small number of data blocks (i.e., D) complemented by a large number of parity blocks (i.e., P) to balance data redundancy and algorithm performance (i.e., theoretical bound). However, a small number of data blocks (i.e., D) will enlarge the file size of each video block (i.e., data block and parity block), which requires more server bandwidth (i.e., impacting the service quality of video servers). In this context, we will evaluate different combinations of D and P on the overall performance of EMS in §5.

4.2 Deadline-aware multi-server selection

Problem reformulation. We also adopt the widely-used *regret* metric to reformulate our problem. However, different from the expression (3) we cannot easily derive the expected optimal super arm $\mathcal{S}^*$ due to the complicate reward expression (2). To this end, we introduce a vector of probability distributions $\boldsymbol{p} = \big\{p(\mathcal{S}), \forall \mathcal{S} \subseteq \mathcal{K}(t), |\mathcal{S}| \geq D\big\}$, where $\mathcal{S}$ refers to a qualified super arm whose size is at least D, and $p(\mathcal{S})$ refers to the probability that the super arm $\mathcal{S}$ will be played. Intuitively, $\sum_{\mathcal{S}} p(\mathcal{S}) = 1$. In this case, the constraint (1) can be rewritten as $\sum_{\mathcal{S}} p(\mathcal{S})|\mathcal{S}| \leq H$. In addition, given any super arm $\mathcal{S}$ and the expectation of local service quality of arm k which is denoted by $\theta_k^* = \mathbb{E}[\theta_k(m)]$, we can derive the expected reward $r^*(\mathcal{S})$ of super arm $\mathcal{S}$ in terms of the expression (2). Then, we can easily solve the following linear problem:

$$\begin{array}{ll} \max & \sum_{\mathcal{S}} p(\mathcal{S}) r^*(\mathcal{S}) \\ \text{s. t.} & \sum_{\mathcal{S}} p(\mathcal{S})|\mathcal{S}| \leq H, \\ & \sum_{\mathcal{S}} p(\mathcal{S}) = 1, \\ \text{var} & p(\mathcal{S}) \in [0, 1], \end{array}$$

to derive the optimal probability $p^*(\mathcal{S})$ for each super arm $\mathcal{S}$. After that, we can define the cumulative regret of any given algorithm as follows:

$$\Delta_{ALG} = M \sum_{\mathcal{S}} p^*(\mathcal{S}) r^*(\mathcal{S}) - \mathbb{E}\Big[\sum_{m=1}^{M} r\big(\mathcal{S}(m)\big) \Big]. \tag{10}$$

To cope with the long-term constraint (1), we first introduce a virtual queue $Q(m)$ with dynamics:

$$Q(m+1) = \max\big\{Q(m) - H,\, 0\big\} + |\mathcal{S}(m)|, \tag{11}$$

and further resort to the Lyapunov drift-plus-penalty technique [43] to derive the reformulated problem for deadline-aware multi-server selection as follows:

$$\begin{aligned} &\min_{\{\mathcal{S}(m), m \geq 1\}} V \Delta_{ALG} + \textstyle\sum_{m=1}^{M} Q(m)|\mathcal{S}(m)| \\ &\Leftrightarrow \max_{\{\mathcal{S}(m), m \geq 1\}} \textstyle\sum_{m=1}^{M} \Big[V r\big(\mathcal{S}(m)\big) - Q(m)|\mathcal{S}(m)| \Big], \end{aligned} \tag{12}$$

where V is a predefined system parameter, which is widely used in the Lyapunov drift-plus-penalty technique.

Algorithm design. According to the above formulation and the expression (2), we can only observe the feedback for each super arm $\mathcal{S}$ as a whole. In this context, we will design a UCB based algorithm for such a combinatorial MAB problem. Specifically, we denote by $\widetilde{\theta}_k(m)$ the UCB estimate of the service quality of arm k at the beginning of the local training round m which is given by

$$\widetilde{\theta}_k(m) \triangleq \min\big\{\theta_k(m-1) + \sqrt{2\ln m / n_k(m-1)},\, 1\big\}, \tag{13}$$

where $n_k(m)$ is updated by the expression (5) and we utilize such a truncated UCB estimate to ensure the positive reward in terms of the reward expression (2). Next, we provide the criterion

Algorithm 2: D-EMS

Input: $\Theta_k(t)$, $\mathcal{K}(t)$, D, H
Output: $\mathcal{S}^*(m), m \in \{1, 2, \ldots, M\}$
▷ *Initial model download from the portal server*
$\theta_k(0) = \Theta_k(t), n_k(0) = 1, k \in \mathcal{K}(t)$;
▷ *RL based local training at the client side*
For each video chunk $m \in \{1, 2, \ldots, M\}$ **do**
  Calculate $\widetilde{\theta}_k(m)$ by the expression (13);
  Sort $\{\widetilde{\theta}_k(m), k \in \mathcal{K}(t)\}$ in the descending order;
  Select the top D arms and repeatedly add the next arm from the sorted order until the new added one cannot increase the objective in (12) as $\mathcal{S}^*(m)$;
  Update $\theta_k(m)$ by the expression (4);
  Update the virtual queue $Q(m)$ by the expression (11);
▷ *Asynchronous global aggregation at the portal server side*
Update $\Theta_k(t+1)$ by $\theta_k(M)$ such as
  $\Theta_k(t+1) = \big[\Theta_k(t) * t + \theta_k(M)\big]/(t+1)$.

of playing a super arm $\mathcal{S}(m)$ in each local training round m to maximize the following objective:

$$\mathcal{S}^*(m) = \underset{\widetilde{\mathcal{S}}(m)\subseteq\mathcal{K}(t)}{\arg\max} \Big[Vr\big(\widetilde{\mathcal{S}}(m)\big) - Q(m)|\widetilde{\mathcal{S}}(m)|\Big]. \quad (14)$$

Here, $r\big(\widetilde{\mathcal{S}}(m)\big)$ substitutes $\theta_k(m)$ in $r\big(\mathcal{S}(m)\big)$ with $\widetilde{\theta}_k(m)$. Although the problem in (14) is a one-shot optimization, it is still hard to handle due to the product of a sequence in $r\big(\widetilde{\mathcal{S}}(m)\big)$. To this end, we introduce the following lemma.

Lemma 1. Without loss of generality, we assume $\widetilde{\theta}_1(m) \geq \widetilde{\theta}_2(m) \geq \cdots \geq \widetilde{\theta}_{k_t}(m)$ in round m. Consider all possible subsets $\widetilde{\mathcal{S}}(m) \subseteq \mathcal{K}(t)$ with a given cardinality c (i.e., $|\widetilde{\mathcal{S}}_c(m)| = c$), then the optimal $\widetilde{\mathcal{S}}_c^*(m)$ satisfying

$$\widetilde{\mathcal{S}}_c^*(m) = \underset{\widetilde{\mathcal{S}}_c(m)\subseteq\mathcal{K}(t)}{\arg\max} \Big[Vr\big(\widetilde{\mathcal{S}}_c(m)\big) - Q(m)|\widetilde{\mathcal{S}}_c(m)|\Big]$$

refers to the top c video servers in terms of the value of $\widetilde{\theta}_k(m)$. In other words, $\widetilde{\mathcal{S}}_c^*(m) = \{\widetilde{\theta}_1(m), \widetilde{\theta}_2(m), \ldots, \widetilde{\theta}_c(m)\}$.

Proof. We prove this lemma by contradiction. Specifically, we consider $\widetilde{\mathcal{S}}_1(m)$ is the optimal subset with cardinality c, and assume there is a server $v \notin \widetilde{\mathcal{S}}_1(m)$ satisfying $\widetilde{\theta}_v(m) \geq \widetilde{\theta}_c(m)$ (i.e., the c-th largest value of $\widetilde{\theta}_k(m), \forall k \in \mathcal{K}(t)$). In this case, there must exist a server $v' \in \widetilde{\mathcal{S}}_1(m)$ satisfying $\widetilde{\theta}_{v'}(m) \leq \widetilde{\theta}_c(m)$. Then, we can build a subset $\widetilde{\mathcal{S}}_2(m) = \widetilde{\mathcal{S}}_1(m)\backslash\{v'\}\cup\{v\}$. In other words, $\widetilde{\mathcal{S}}_2(m) \setminus \{v\} \equiv \widetilde{\mathcal{S}}_1(m) \setminus \{v'\}$. According to the specific expression in (2), we can verify that $r\big(\widetilde{\mathcal{S}}_2(m)\big) - r\big(\widetilde{\mathcal{S}}_1(m)\big) \geq 0$, which produces a contradiction (i.e., $\widetilde{\mathcal{S}}_1(m)$ is not optimal).

In terms of the above lemma, we can derive $\mathcal{S}^*(m)$ by sorting the arms in the descending order of their UCB estimates, selecting the top D arms and adding the next arm from the sorted order repeatedly until the new added one cannot increase the objective in (12). The detailed algorithm D-EMS are given in Alg. 2.

Performance analysis. To begin with, we can prove that the above deadline-aware multi-server selection is feasibility (i.e., the long-term constraint (1) is satisfied). Then, its complexity is also lightweight (i.e., dominated by the sorting). Besides, the expected cumulative regret can be upper bounded by

$$\Delta_{ALG} \leq \frac{M(H+K_t)^2}{2V} + \frac{\pi^2}{6}K_t \leq \frac{(H+K_t)^2}{2}\sqrt{M} + \frac{\pi^2}{6}K_t,$$

where K_t refers to the size of $\mathcal{K}(t)$. The detailed proof is tedious, and therefore we provide it in our online technical report [42].

Note that if we give V a reasonably large value (i.e., $V \geq \sqrt{M}$), then the cumulative regret grows as $\mathcal{O}(\sqrt{M})$ which is in a sub-linear way in the number of local training rounds (i.e., the number of video chunks). We will evaluate different values of V on the overall performance of EMS in §5.

5 Performance Evaluation

In this section, we implement a prototype of EMS and conduct extensive testbed driven experiments to evaluate the performance of the proposed multi-server selection algorithms.

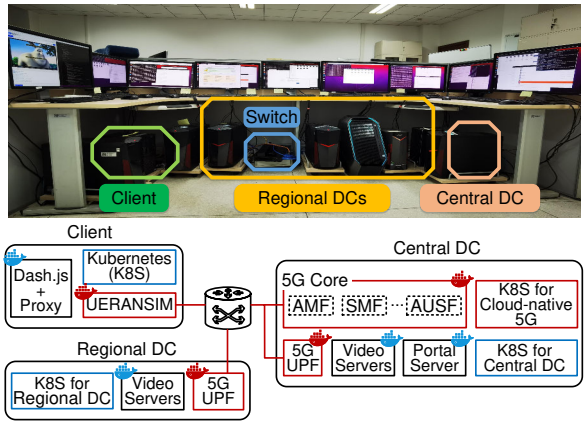


Fig. 6: The testbed of EMS.

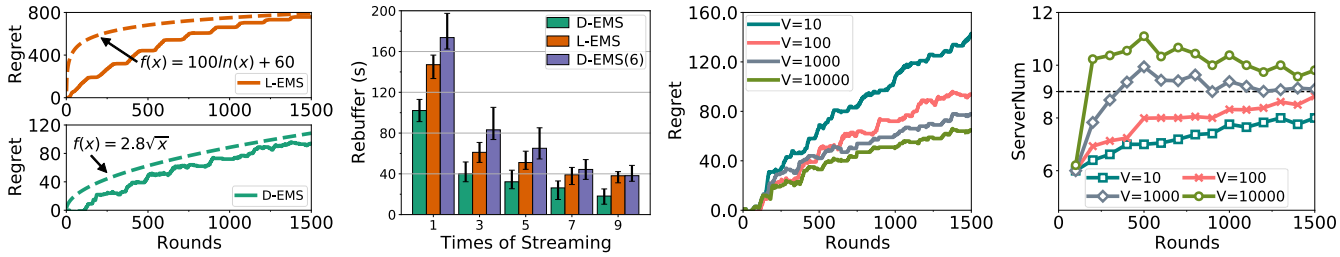
5.1 Testbed implementation

As shown in Fig. 6, our testbed consists of 9 desktops connected with a Huawei gigabit switch, and each desktop will exploit Kuberenets (K8S) to orchestrate the containerized 5G components as well as video clients and servers. Particularly, K8S for Cloud-native 5G orchestrates all the containers in the regional DCs and central DC dedicated for 5G components (i.e., 5G UPF and 5G core), and K8S for Central (Regional) DC orchestrates local containers for edge services (e.g., edge computing and caching).

Cloud Native 5G network. We exploit two open-source projects UERANSIM [44] and OPEN5GS [45] to emulate the components and functionalities of 5G, in which UERANSIM works as 5G UE and RAN (gNodeB) and OPEN5GS works as 5G UPF and 5G Core. In the testbed, we launch one K8S in the central DC to manage the cloud native 5G network. That is, it will create a UERANSIM container for each video client, a 5G UPF container for each regional DC, and a 5G UPF and a 5G Core container for the central DC. These 5G components cooperate harmoniously with a specific configuration file, and they provide the underlying connection for each video client and server.

Video Client. For ease of implementation, we create a container to emulate each video client that consists of a video player in the browser (i.e., dash.js [46]) and a proxy created by Python that enables multi-server selection and erasure-coded chunk decoding. Specifically, we exploit Zfec [47], an open-source Python package for erasure coding, overwrite Zfec's APIs to achieve in-memory decoding (i.e., avoiding the data transmission between memory and storage), and create a local socket program to achieve the data transmission between Zfec and dash.js.

Video Server. We create a container to emulate each video server that is built by Apache Tomcat [48]. As to erasure-coded



(a) Algorithm performance. (b) Streaming performance. (c) D-EMS regret with different V. (d) Number of selected servers with different V under D-EMS.

Fig. 7: Performance of the proposed multi-server selection algorithms (i.e., L-EMS and D-EMS).

video, we download an 8K video (10-minute length) from Youtube and encode it with the bitrate 100Mbps by FFmpeg [49]. Then, we divide the encoded video into multiple video chunks with a typical 4-second duration (i.e., 150 video chunks in total) and package all the derived chunks in the DASH format by MP4Box [50]. We erasure-code each video chunk into D data blocks and P parity blocks by Zfec [47] ($D=P=6$ by default), and deploy one video block on one video server[7].

Bandwidth setting. We synthesize a series of downstream bandwidth samples between client and server in terms of the real data distribution in Fig. 3. Specifically, we exploit the curve fitting tool in Matlab to derive three approximate distributions for each CDF curve (i.e., Gamma, Burr and Johnson) and then randomly generate 10000 bandwidth samples denoted by $\boldsymbol{x}$ for each distribution (AVG refers to the average value of those samples). Next, we consider three classes of bandwidth settings (i.e., Low: 12 – 15Mbps, Mid: 18 – 21Mbps and High: 24 – 27Mbps), indicating the mean value denoted by MEAN of synthetic bandwidth used in the following evaluation. Note that these values are intentionally selected in terms of the given video bitrate and erasure-coding pattern mentioned above, and different value settings have little impact on the performance evaluation (e.g., the comparison among different algorithms). In this context, we can derive the final bandwidth sample x' from any a generated sample $x \in \boldsymbol{x}$ by a simple transformation (Other transformations are also applicable):

$$x' = x \times (\mathsf{MEAN} - \mathsf{MIN})/\mathsf{AVG} + \mathsf{MIN},$$

where MIN refers to the specified minimum bandwidth, which is set to 5Mbps for simplicity. In the evaluation, we exploit the Linux built-in Traffic Control (TC) tool in each server to control its bandwidth capacity every 2 seconds[8], in terms of the derived final bandwidth samples (i.e., $\boldsymbol{x}'$).

5.2 Testbed driven evaluation

Metric. We exploit the cumulative regret denoted by Regret to indicate algorithm performance (i.e., the cumulative gap between the expected optimal reward and the actual reward derived by a given algorithm), and exploit the total rebuffer time denoted by Rebuffer to indicate streaming performance since our experiments do not involve any ABR algorithms (i.e., video bitrate is fixed). Note that the expected optimal reward can be easily calculated in terms of the given system setting as mentioned in §4, the actual reward can be obtained when D video blocks of each chunk are downloaded, and the total rebuffer time can be obtained at the end of each time of streaming. We conduct the performance evaluation by answering the following questions.

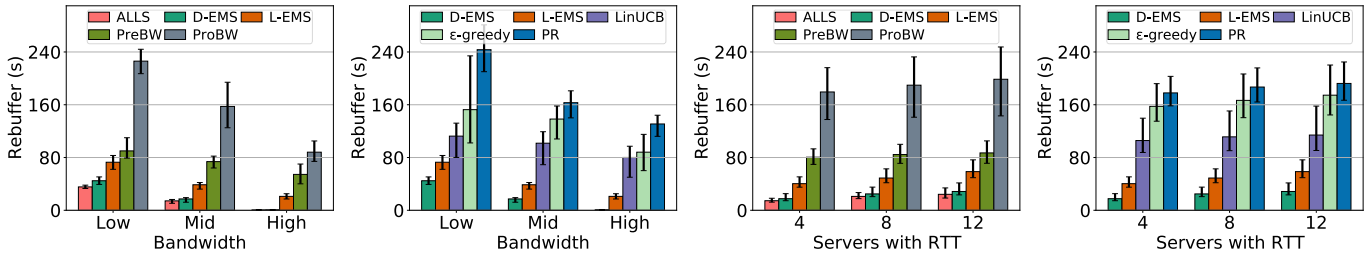
Q1: Does the practical performance of the proposed algorithms coincide with the theoretical analysis?

Setting. The experiment consists of 1 client and 12 servers that store 6 data blocks and 6 parity blocks (i.e., erasure-coding pattern is 6/6). The client streams the test video 10 times (i.e., $150 \times 10 = 1500$ algorithm execution rounds). One third of servers respectively select Low, Mid and High class associated with a randomly selected bandwidth distribution from Gamma, Burr and Johnson to generate 3000 bandwidth samples (i.e., one algorithm execution round corresponds to a 4s video chunk, and the bandwidth is changed every 2s). As to D-EMS, we set the predefined system parameter $V = 100$ and the threshold of the average number of selected servers $H = 9$. Besides, we consider D-EMS(6) in which the parameter $V = 100$ and the threshold $H = 6$, so as to facilitate the performance comparison between D-EMS and L-EMS. Without specific statements, the following experiments will adopt the same setting.

Result. The evaluation results are shown in Fig. 7. To begin with, we can find from Fig. 7(a) that the achievable regret of each algorithm is consistency with the theoretical analysis. That is, it is in a sub-linear way in the number of algorithm execution rounds. In addition, Fig. 7(b) reveals that the total rebuffer time of one time of streaming achieved by each algorithm decreases with algorithm execution rounds increasing. For example, the total rebuffer time of the 9-th streaming under L-EMS and D-EMS respectively achieves 74% and 82% reduction compared with that of the 1-th streaming. These results indicate that our proposed algorithms can effectively filter out good servers and accordingly achieve better performance over time. In addition, we can find that the performance of L-EMS is better than that of D-EMS(6) (i.e., smaller total rebuffer time and smaller fluctuation). For example, the total rebuffer time of the former one is 15% smaller than the latter one's at the 1-th streaming round. This gap is shorten with the streaming round increasing. In other words, L-EMS has a faster convergence rate compared with D-EMS when both of them select the minimum number of servers (i.e., 6 in the current setting) for chunk recovery. The reason is that L-EMS by definition aims to filter out the "best" video servers while D-EMS will only filter out the "qualified" ones. We also evaluate the performance of D-EMS with different V as shown in Fig. 7(c) and Fig. 7(d), which are in accordance with the behavior of

7. Note that different 8K videos have little impact on the performance evaluation by using the above erasure-coded video making. In addition, we do not consider multiple bitrate versions so as to eliminate the impact of adaptive bitrate algorithms (i.e., we do not involve any ABR algorithms).

8. This setting is used to generate various and dynamic network conditions, and different settings have little impact on the algorithm comparison.



(a) Multi-source streaming. (b) UCB based algorithms. (c) Multi-source streaming. (d) UCB based algorithms.

Fig. 8: Performance comparison under different algorithms.

Lyapunov-based algorithms. That is, a larger V can facilitate the objective while loosing the long-term constraint.

Q2: How is the performance of the proposed algorithms compared with alternative ones?

Setting. The experiment considers both multi-source streaming algorithms and UCB based algorithms for performance evaluation. The first one includes

- *ALLS*: adopt the same setting with EMS in which the client will download the video blocks from all the servers;
- *PreBW* [13], [32]: adopt the replication storage mode (i.e., only two servers store the whole video as our erasure-coding pattern is 6/6) in which the client will proportionally download the video chunk in terms of the bandwidth prediction (i.e., Harmonic mean method);
- *ProBW*: also adopt the replication storage mode in which the client will proportionally download the video chunk in terms of the bandwidth probing [31].

The second one includes

- *LinUCB* [51]: a contextual-aware bandit algorithm which picks one arm per round. Here, the context is the predicted bandwidth and we run it D times to get sufficient servers;
- *ϵ-greedy* [14]: attempt to select D servers by using a variant of ϵ-greedy algorithm;
- *PR*: regard the UCB estimate as the probability to conduct the multi-server selection. It will repeatedly operate until D servers are selected.

We consider two kinds of experiments: (1) only varying transmission bandwidth; (2) given the transmission bandwidth while vary propagation delay. For the first one, we consider all the 12 servers will select the same class from Low, Mid or High associated with the Gamma distribution and do not take the round-trip time (RTT) into account. For the second, we consider all the 12 servers will select the same class from Mid associated with the Gamma distribution and respectively control RTT values for 4, 8 and 12 servers. Specifically, we consider the latest broadband measurement dataset (2022.01 – 2022.06) released by federal communications commission (FCC) [52], which contains a "Latency Under Load" field to indicate the round-trip time between clients and servers. We derive the CDF of RTT values from the dataset and exploit it to adjust the transmission delay for servers. Note that since the round-trip time is impacted by many factors (e.g., packet queueing and loss in the routers) in practice, we instead adjust the transmission delay at the server side to simulate the round-trip time. That is, we exploit the Linux built-in traffic control tool in each server to control its transmission delay every 2 seconds, in terms of the derived CDF of RTT values from the FCC dataset. We repeatedly run each experiment 20 times.

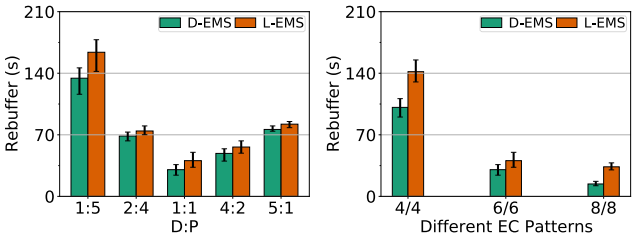
Results. Here, we consider the streaming performance under different algorithms. Fig. 8 presents the total rebuffer time of the 10-th streaming. In the first experiment (i.e., Fig. 8(a) and Fig. 8(b)), L-EMS and D-EMS can achieve better performance. For example, when the bandwidth setting is the Mid class for servers, the total rebuffer time under L-EMS is on average 61% and 73% reduction compared with the multi-source streaming algorithms (except for *ALLS*) and the UCB based algorithms, respectively. The reasons are two-fold. First, it is difficult to accurately predict or probe bandwidth, resulting in suboptimal multi-source streaming. Second, the comparable UCB based algorithms lack theoretical performance, which could not always select reasonably good servers per round and result in suboptimal performance over time. In addition, the performance of D-EMS is much better (i.e., 63% on average) than that of L-EMS. The main reason is that D-EMS associated with the long-term constraint can exploit and explore more servers per round, which contributes to filtering out qualified servers. Indeed, as shown in Fig. 8(a) the performance of D-EMS is only on average 13% worse than that of *ALLS* (Low: 20%, Mid: 17% and High: 3%) In the second experiment (i.e., Fig. 8(c) and Fig. 8(d)), we can find that the algorithm comparison shares the same trend when taking RTT into account. The reason is that the operation of all the algorithms (except ALLS) depends on the end-to-end bandwidth, which is mainly influenced by the allocated server bandwidth and the propagation delay (i.e., RTT) in practice, and the impact of RTT on the end-to-end bandwidth could be viewed as a "penalty" for the allocated server bandwidth. Indeed, when the number of servers with RTT increase, the performance of all the algorithms will slightly decrease.

As to the algorithm running time, we should emphasize that all the algorithms are lightweight compared with the duration of video chunk (i.e., 4s). For example, the running time of *LinUCB* and *ϵ-greedy* are less than 1ms, and that of *PreBW* is less than 20ms. Although our proposed D-EMS has a longer running time, it is still acceptable in practice as discussed in Q5.

Q3: How is the impact of erasure-coding pattern (i.e., different combinations of D and P) on the performance of the proposed algorithms?

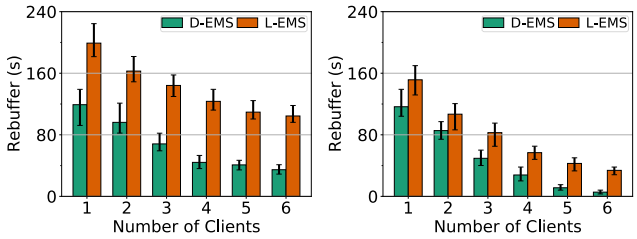
Setting. We consider two kinds of experiments. The first one fixes the number of video blocks (i.e., $D+P=12$) while changing the proportion of D and P. The second one considers a different number of video blocks while keeping the proportion of D and P (i.e., 1:1). As to the bandwidth setting, we consider that all the servers will select the Mid class associated with the Gamma distribution. Both experiments are run 20 times.

Results. Here, we consider the streaming performance with



(a) Different proportions. (b) Different EC patterns.

Fig. 9: The impact of erasure-coding (EC) pattern on the performance.



(a) Low bandwidth setting. (b) Mid bandwidth setting.

Fig. 10: Performance comparison with different number of clients.

different erasure-coding patterns. Fig. 9 presents the total rebuffer time of the 10-th streaming. As shown in Fig. 9(a), the total rebuffer time experiences a quick decrease in the beginning and a slight increase in the end with the number of data blocks (i.e., D) increasing. This observation verifies our argument in §4.1. That is, a small number of data blocks will enlarge the file size of each video block and accordingly require more server bandwidth. In contrast, a large number of data blocks will enlarge the upper bound of regret, both of which could damage the practical performance. Therefore, we consider that EMS should exploit a relatively balanced combination of D and P. As shown in Fig. 9(b), the total rebuffer time decreases from the pattern 4/4 to 6/6, while keeping it relatively stable from 6/6 to 8/8. The reasons are two-fold. First, The larger number of video blocks refers to a larger number of video servers, which indicates the more bandwidth at the server side and accordingly benefits the streaming performance. Second, the streaming performance cannot continuously increase with the number of video blocks increasing, since decoding more video blocks require more time as shown in Fig. 11.

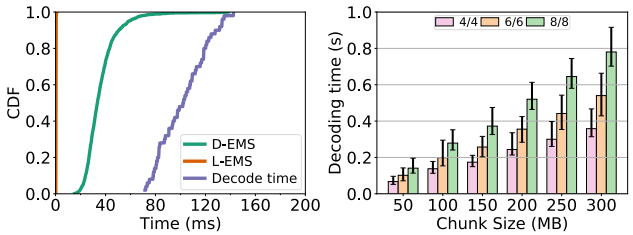
Q4: Does the federated learning paradigm benefit EMS?

Setting. From the previous evaluations, we can find the superior performance of the RL-based multi-server selection. Here, we wonder if the global aggregation can further enhance the performance. To this end, this experiment considers multiple clients to conduct video streaming simultaneously, consisting of 6 clients and 12 servers. Note that the setting where only one client conducts video streaming represents the no FL scenario (i.e., single user scenario). The erasure-coding pattern is still 6/6. As to the bandwidth setting, we consider that all the 12 servers will select the same class from Low or High associated with the Gamma distribution. Note that as our bandwidth samples are synthesized from real data distributions, we do not consider the server bandwidth competition from multiple clients[9]. In other words, we exploit the TC tool to control the server bandwidth capacity for each pair of client and server. We repeatedly run the experiment 20 times.

Results. Here, we consider the streaming performance with a different number of clients. Fig. 10 presents the total rebuffer time of the 10-th streaming. Intuitively, the performance of both proposed algorithms gets better as the number of concurrent clients increases. For example, the performance of D-EMS when the number of clients is 6 is respectively 94% and 42% better than that when the number of clients is 1 (i.e., no FL scenario) in the Mid and Low bandwidth setting. This is because given a set of clients, increasing the number of concurrent clients is approximately equivalent to increasing the times of streaming. Therefore, we can hold that the federated learning paradigm, including RL-based multi-server selection and global aggregation, can achieve a good performance.

Q5: How is the overhead of EMS?

Setting. At last, we evaluate the extra overhead of EMS. Compared with the general video streaming, EMS as discussed in §3 introduces multi-server selection, video chunk decoding and transmission of local trained service quality of servers. To this end, we run the experiment in Q1 multiple times to collect the algorithm running time and chunk decoding time. Besides, we further evaluate the chunk decoding time with different system settings such as erasure-coding patterns and chunk sizes. Note that we do not need to consider the transmission of local trained service quality of servers, since it is of small size and only occurs when the video streaming finishes.



(a) CDF of algorithm running time and chunk decoding time. (b) Chunk decoding time with different system settings.

Fig. 11: System overhead.

Results. The evaluation results are shown in Fig. 11. From Fig. 11(a), we can observe that EMS is a lightweight framework. For example, the running time of both L-EMS and D-EMS is less than 70 ms, and the erasure-coding based chunk decoding time is 100 ms on average, which is negligible compared with the 4s chunk duration. From Fig. 11(b), we can find that the erasure-coding pattern greatly impacts the decoding time. For example, the decoding time when the erasure-coding pattern is 8/8 is on average 54% longer than that when it is 4/4, and it will exceed 1s when the chunk size is 300MB (i.e., bitrate is $300\times8/4$=600Mbps). However, we should emphasize that a fine-granularity erasure-coding pattern also means many cooperative video servers, which can greatly reduce the data transmission time. In this context, we believe multi-source streaming and erasure-coding storage could complement each other in practice, which indicates the feasibility of EMS.

We also exploit the NVIDIA Jetson TX2 [53] (Quad-Core ARM Cortex-A57 CPU and 8G RAN) to emulate a mid-end

9. In practice, video servers can exploit some isolation techniques to allocate bandwidth for users so as to avoid resource competition. For example, as to each video, they can set many backup containers with the same amount of resources and serve each client with one container exclusively.

mobile device and repeat the above experiment. The derived algorithm running time and chunk decoding time are slightly longer than the test desktop's (Dual-Core INTEL i7-8700 CPU and 8G RAN). For example, the running time of both L-EMS and D-EMS is less than 120 ms, and the erasure-coding based chunk decoding time is 210 ms on average (i.e., roughly 0.3s overhead in total). Therefore, we consider that EMS is lightweight and can be applied to many off-the-shelf mobile devices in practice.

6 Discussion and Limitation

EMS is a practical and promising framework built on top of cloud native 5G networks. As containers and microservices can offer many benefits such as agility, flexibility, resilience and cost efficiency, cloud native 5G networks have been receiving increasing attention from many MNOs and will be commercially available in the near future [5], [6]. As shown in Fig. 1, EMS can be viewed as a specific "microservice" tailored to UHD video streaming, in which each video server consists of a video container and several backup containers. Besides, as EMS operates at the application layer, it does not need to care about but benefits from various promising underlying network technologies such as 5G new radio specifications, flexible ethernet and segment routing.

EMS can also be extended to support provider-driven multi-server selection. According to the algorithm design in §4, the client-driven multi-server selection is conducted by the individual user with the granularity of video chunk (i.e., local control), aiming to maximize user QoE while respecting system cost constraints. With a different design philosophy, the provider-driven multi-server selection will be conducted by the portal server with the granularity of time slot (i.e., global control), aiming to minimize system cost while respecting user QoE constraints. We consider the proposed federated learning based service quality of video servers can also benefit the provider-driven multi-server selection, such as facilitating the modeling of average user QoE per user area, and we can design a deep reinforcement learning based algorithm for the multi-server selection, due to the sufficient processing capacity of the portal server.

Realistic evaluation issue. We should emphasize that our testbed tries the best to emulate the realistic video streaming for EMS, since (1) it controls the transmission bandwidth of each video server for each client; (2) it exploits the end-to-end bandwidth fluctuation from real datasets and measurements; (3) it takes UERANSIM and OPEN5GS, two open-source 5G projects into account so as to emulate the behavior of cloud native 5G network. Therefore, we believe our evaluation results are meaningful. Nevertheless, due to budget and hardware limit our testbed driven evaluation is an emulational and small-scale laboratory experiment (e.g., no packet loss or queueing at the routers), and we will consider a more realistic and large-scale evaluation involving 5G network and edge servers through the cooperation with some mobile network operators in the future.

User mobility issue. In this paper, we simply assume user movement during video streaming should be relatively small. In other words, the event that a user moves from one area to another during video streaming occurs infrequently. Therefore, if EMS detects the event occurrence (e.g., with the help of the functionalities of 5G Core), it will simply exclude that user from the global aggregation of service quality. More sophisticated processing for user area handover in EMS will be considered in the future work.

7 Conclusion

We present EMS, a novel UHD video streaming framework within cloud native 5G networks by integrating erasure-coded storage with multi-source streaming. We respectively introduce a deadline-aware and a latency-sensitive metric to indicate the service quality of video servers and advocate a federated learning paradigm for the adaptive service quality update, including a reinforcement learning based multi-server selection and a global service quality update. We cast the multi-server selection associated with the restriction on the average number of selected servers per video chunk into two kinds of MAB models in terms of the proposed service quality metrics and correspondingly design lightweight UCB based algorithms with a theoretical performance guarantee. Extensive testbed driven experiments confirm the superiority of the proposed multi-server selection algorithms.

References

[1] S. Aggarwal, S. Paul, P. Dash, *et al.*, "How to evaluate mobile 360-degree video streaming systems?," in *ACM HotMobile*, 2020.

[2] B. Han, Y. Liu, and F. Qian, "Vivo: Visibility-aware mobile volumetric video streaming," in *ACM MOBICOM*, 2020.

[3] "MIIT: China has 260M 5G subs; telecom business revenue significantly increased." Available in: https://techblog.comsoc.org/tag/china-telecom/.

[4] B. Zolfaghari, G. Srivastava, S. Roy, *et al.*, "Content delivery networks: state of the art, trends, and future roadmap," *ACM Computing Surveys (CSUR)*, 2020.

[5] "Reimagining the end-to-end mobile network in the 5G era." Available in: https://www.cisco.com/c/en/us/solutions/service-provider/mobile-internet/reimagining-mobile-network.html.

[6] "5G cloud-native infrastructure." Available in: https://www.f5.com/pdf/solution-guides/deploying-cloud-native-infra-and-5g-core-overview.pdf.

[7] J. Kwak, L. B. Le, G. Iosifidis, K. Lee, and D. I. Kim, "Collaboration of network operators and cloud providers in software-controlled networks," *IEEE Network*, 2020.

[8] A. Narayanan, E. Ramadan, R. Mehta, *et al.*, "Lumos5g: Mapping and predicting commercial mmwave 5g throughput," in *ACM Internet Measurement Conference (IMC)*, 2020.

[9] L. Pu, L. Jiao, X. Chen, *et al.*, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, 2018.

[10] "Huawei's practice in migration to cloud native based 5G telco cloud." Available in: https://www.openstack.org/videos/summits/virtual/Huaweis-Practice-in-Migration-to-Cloud-Native-based-5G-Telco-Cloud.

[11] Y. Zhang, Y. Zhang, *et al.*, "Improving quality of experience by adaptive video streaming with super-resolution," in *IEEE INFOCOM*, 2020.

[12] A. Zhang, C. Wang, X. Liu, *et al.*, "Mobile volumetric video streaming enhanced by super resolution," in *ACM MOBISYS*, 2020.

[13] S. Da Silva, S. Ben Mokhtar, S. Contiu, *et al.*, "Privatube: Privacy-preserving edge-assisted video streaming," in *ACM/IFIP Middleware*, 2019.

[14] X. Chen, M. Zhao, X. Yang, Z. Li, *et al.*, "The cask effect of multi-source content delivery: Measurement and mitigation," in *IEEE ICDCS*, 2019.

[15] K. Rashmi, M. Chowdhury, J. Kosaian, I. Stoica, and K. Ramchandran, "Ec-cache: Load-balanced, low-latency cluster caching with online erasure coding," in *USENIX OSDI*, 2016.

[16] A. O. Al-Abbasi and V. Aggarwal, "Video streaming in distributed erasure-coded storage systems: Stall duration analysis," *IEEE/ACM Transactions on Networking (ToN)*, 2018.

[17] M. Uluyol, A. Huang, A. Goel, M. Chowdhury, and H. V. Madhyastha, "Near-optimal latency versus cost tradeoffs in geo-distributed storage," in *USENIX NSDI*, 2020.

[18] "ACM MMSys grand challenge on bandwidth estimation for real-time communications." Available in: https://2021.acmmmsys.org/rtc_challenge.php.

[19] "ACM MM grand challenge on meet deadline requirements." Available in: https://www.aitrans.online/MMGC2021/.

[20] H. Wang, K. Wu, J. Wang, and G. Tang, "Rldish: Edge-assisted qoe optimization of http live streaming with reinforcement learning," in *IEEE INFOCOM*, 2020.

[21] F. Y. Yan, H. Ayers, C. Zhu, *et al.*, "Learning in situ: a randomized experiment in video streaming," in *USENIX NSDI*, 2020.
[22] H. Wang, F. Xu, Y. Li, *et al.*, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," in *ACM IMC*, 2015.
[23] E. A. Walelgne, A. S. Asrese, J. Manner, *et al.*, "Clustering and predicting the data usage patterns of geographically diverse mobile users," *Elsevier Computer Networks*, 2021.
[24] "Internet connection speed recommendations on netflix." Available in: https://help.netflix.com/en/node/306.
[25] "Puffer: a video streaming dataset from stanford university." Available in: https://puffer.stanford.edu/data-description/.
[26] "Beyond throughput, the next generation: a 5g dataset with channel and context metrics." Available in: https://github.com/uccmisl/5Gdataset.
[27] "Chromedriver: Webdriver for chrome." Available in: https://chromedriver.chromium.org/getting-started/getting-started---android.
[28] X. Yin, A. Jindal, *et al.*, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *ACM SIGCOMM*, 2015.
[29] A. Bentaleb, B. Taani, A. C. Begen, *et al.*, "A survey on bitrate adaptation schemes for streaming media over HTTP," *IEEE Communications Surveys & Tutorials*, 2018.
[30] D. Jurca, J. Chakareski, J.-P. Wagner, and P. Frossard, "Enabling adaptive video streaming in P2P systems," *IEEE Communications Magazine*, 2007.
[31] "Bitswap: the core module of ipfs for exchanging blocks of data." Available in: https://docs.ipfs.io/concepts/bitswap/.
[32] Y.-C. Chen, D. Towsley, and R. Khalili, "Msplayer: Multi-source and multi-path video streaming," *IEEE Journal on Selected Areas in Communications (JSAC)*, 2016.
[33] A. Badita, P. Parag, and J.-F. Chamberland, "Latency analysis for distributed coded storage systems," *IEEE Transactions on Information Theory (TIT)*, 2019.
[34] K. Shanmugam, N. Golrezaei, A. G. Dimakis, *et al.*, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory (TIT)*, 2013.
[35] V. Bioglio, F. Gabry, and I. Land, "Optimizing mds codes for caching at the edge," in *IEEE GLOBECOM*, 2015.
[36] X. Xu and M. Tao, "Modeling, analysis, and optimization of coded caching in small-cell networks," *IEEE Transactions on Communications (TOC)*, 2017.
[37] X. Wu, Q. Li, V. C. Leung, and P.-C. Ching, "Joint fronthaul multicast and cooperative beamforming for cache-enabled cloud-based small cell networks: An mds codes-aided approach," *IEEE Transactions on Wireless Communications (TWC)*, 2019.
[38] B. Alt, T. Ballard, *et al.*, "CBA: Contextual quality adaptation for adaptive bitrate video streaming," in *IEEE INFOCOM*, 2019.
[39] S. Boldrini, L. De Nardis, *et al.*, "muMAB: A multi-armed bandit model for wireless network selection," *Algorithms*, 2018.
[40] A. Hodroj, M. Ibrahim, *et al.*, "Enhancing dynamic adaptive streaming over http for multi-homed users using a multi-armed bandit algorithm," in *IEEE IWCMC*, 2019.
[41] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.
[42] "Online technical report." Available in: https://www.dropbox.com/s/dnz6oxj3n8dl8rq/TMC2022_technicalreport.pdf?dl=0.
[43] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, 2010.
[44] "Ueransim: Open source 5g ue and ran (gnodeb) implementation.." Available in: https://github.com/aligungr/UERANSIM/wiki.
[45] "Open5gs: Open source implementation for 5g core and epc.." Available in: https://open5gs.org/.
[46] "dash.js: A reference client implementation for the playback of MPEG DASH via javascript and compliant browsers." Available in: https://github.com/Dash-Industry-Forum/dash.js/wiki.
[47] "Zfec: this package implements an erasure code." Available in: https://tahoe-lafs.org/trac/zfec.
[48] "Apache tomcat: an open source implementation of the java servlet and websocket technologies." Available in: http://tomcat.apache.org/.
[49] "FFmpeg, a complete, cross-platform solution to record, convert and stream audio and video." https://ffmpeg.org/.
[50] "MP4Box, a multi-purpose mp4 file manipulation." https://gpac.wp.imt.fr/tag/mp4box/.
[51] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *WWW*, 2010.
[52] "Measuring broadband raw data releases." Available in: https://www.fcc.gov/oet/mba/raw-data-releases.
[53] "Technical specifications of jetson tx2 module." https://developer.nvidia.com/embedded/jetson-tx2.

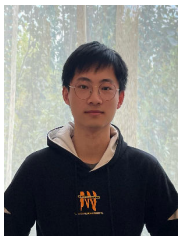
Lingjun Pu is an Associate Professor with Nankai University, Tianjin, China. He received the Ph.D. degree from Nankai University in 2016, and was a joint Ph.D. student with University of Göttingen, Germany, from 2013 to 2015. His current research interest includes programmable networks, edge intelligence, UHD video streaming and resource scheduling.

Jianxin Shi is currently pursuing the Ph.D degree at the College of Computer Science, Nankai University, Tianjin, China. His current research interest includes neural-enhanced video streaming, multimedia content processing and edge intelligence.

Xinjing Yuan is currently pursuing the Ph.D degree at the College of Computer Science, Nankai University, Tianjin, China. She received the Best Paper Award of 2021 IEEE Wireless Communications and Networking Conference (WCNC). Her current research interest includes distributed learning, panoramic video streaming and edge computing.

Xu Chen Xu Chen is a Full Professor with Sun Yat-sen University, Guangzhou, China, Director of Institute of Advanced Networking and Computing Systems, and the Vice Director of National Engineering Research Laboratory of Digital Homes. He received the Ph.D. degree in information engineering from the Chinese University of Hong Kong in 2012, and worked as a Postdoctoral Research Associate at Arizona State University, Tempe, USA, from 2012 to 2014, and a Humboldt Scholar Fellow at the Institute of Computer Science of the University of Goettingen, Germany from 2014 to 2016. He received the prestigious Humboldt research fellowship awarded by the Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young Scientist Runner-up Award, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young Professional Best Paper Award, 2020 IEEE Computer Society Best Paper Awards Runner-Up, Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics (ISI), Best Paper Runner-up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and Best Paper Award of 2017 IEEE International Conference on Communications (ICC). He is currently an Area Editor of the IEEE OPEN JOURNAL OF THE Communications Society, an Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE INTERNET OF THINGS JOURNAL and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Series on Network Softwarization and Enablers.

Lei Jiao received the Ph.D. degree in computer science from the University of Göttingen, Germany. He is currently an assistant professor at the Department of Computer Science, University of Oregon, USA. Previously he worked as a member of technical staff at Nokia Bell Labs in Dublin, Ireland and as a researcher at IBM Research in Beijing, China. He is interested in the mathematics of optimization, control, learning, and economics applied to computer and telecommunication systems, networks, and services. He publishes papers in journals such as JSAC, ToN, TPDS, TMC, and TDSC, and in conferences such as INFOCOM, MOBIHOC, ICNP, ICDCS, SECON, and IPDPS. He is an NSF CAREER awardee. He also received Best Paper Awards of IEEE LANMAN 2013 and IEEE CNS 2019. He was on the program committees of many conferences, including INFOCOM, MOBIHOC, ICDCS, IWQoS, and WWW, and was also the program chair of multiple workshops with INFOCOM and ICDCS.



Tian Zhang is currently pursuing the master's degree at the College of Computer Science, Nankai University, Tianjin, China. His current research interest includes video streaming, resource scheduling and edge intelligence.

Jingdong Xu is a Full Professor with Nankai University, Tianjin, China. She is the Head of the Computer Networks and Information Security Lab. Her research interest includes mobile computing, network security, internet of things and blockchain.