A Large-Scale Measurement and Optimization of Mobile Live Streaming Services

Zhenyu Li, Jinyang Li, Qinghua Wu, Gareth Tyson, and Gaogang Xie

Abstract—Mobile Live Streaming (MLS) services are one of the most popular types of mobile apps. They involve a (often amateur) user broadcasting content to a potentially large online audience via unreliable networks. Nevertheless, we still lack a deep understanding of MLS user behavior that is critical for optimizing MLS systems, despite some active measurements on viewer-side behavior. Using detailed logs obtained from a major MLS provider, this paper first conducts an in-depth measurement study of both viewer-side and broadcaster-side behavior. Key findings include large wasteful uploads, strong viewing locality, and traffic dominance of loyal viewers. Specifically, 33.3% of uploads go unwatched, and the viewership of broadcasters tends to be localized. Inspired by our findings, we propose EDGEOPT– a centralized control center for MLS services for optimizing both the first-mile and the last-mile transmission in MLS. Specifically, EDGEOPT reduces wasteful uploading by 71% through adaptive uploading and enhances the replay quality of popular video segments by 10% via highlights retransmission. EDGEOPT also uses a learning-based content pre-fetching scheme that boosts the viewing startup by 29.5% and offloads at most 80% of the viewing workload from the edge servers with peer-assisted delivery.

Index Terms—Mobile live streaming; User behavior; Edge-side optimization; QoE improvement.

1 INTRODUCTION

The majority of Internet traffic is now video [2], and with the development of mobile devices, Mobile Live Streaming (*MLS*) services like Facebook Live [5], and Periscope [8] have become important contributors. In MLS, anyone can be a broadcaster, streaming video anywhere from a mobile device, which may reach a potentially large audience. This not only removes the traditional live broadcaster's reliance on hardware devices (*e.g.,* computers and cameras) reducing the threshold for starting a live channel and improving the service, but also helps build an era of "live broadcast for the people".

MLS services differ from the traditional live streaming services in that both amateur and professional users can be broadcasters in MLS services, while the contents in traditional live streaming services are often professionally produced (*e.g.,* sports events). This results in several unique challenges in designing MLS systems. First, as MLS broadcasters are often amateurs, both the last and first mile are often unreliable (*e.g.,* 3G/4G), thereby creating potential consequences for overall Quality of Experience (QoE) [33]. This challenge is exacerbated by the unpredictable nature of user-generated uploads, making capacity management more difficult. For instance, although any user can broadcast, their popularity is highly diverse, with some videos going unwatched, whilst others gain huge traction [35]. Second, both broadcasts and views are much shorter than traditional live streaming [41]. Hence, viewers are less patient in waiting for the start of playback [20]. Third, MLS services often offer time-shifted replay views of broadcasts. As replay views are in the form of VoD (video-on-demand), they often require a higher video quality than the live streaming views [36]. Nevertheless, this contradicts live streaming, where low latency is the first priority [41].

To address these challenges and streamline MLS systems, we need a deep understanding of the user behavior in practical MLS services. Existing active measurements of MLS services mostly focus on the infrastructure of MLS platforms [35], [41] (like the CDNs), or the popularity of broadcasters [35]. Nevertheless, they do not provide a detailed analysis of both the broadcasters' and the viewers' behavior in terms of the network resource usage, video QoE and viewership characteristics, mostly because of the limitation of datasets.

With the above in mind, we have obtained 10 days of service logs (1.8TB) from a major MLS service. In contrast to prior works, our data covers *both* the first and the last mile information. The dataset includes 1.9M live broadcasts, 0.4M broadcasters, and 300M views from 2M viewers. In this paper, we focus on quantifying the challenges faced in MLS systems, and evaluating a set of optimizations to streamline the provision of MLS workloads. Specifically, we analyze the user behavior of both broadcasters and viewers first and identify significant volumes of wasteful live video uploads (33.3% of the total upstream traffic). We then proceed to dive into the reasons behind the wastage. We next investigate the geographical popularity of broadcasts to examine the viewership locality in the wild and the impact on video

- *Z. Li, J. Li, and Q. Wu are with the Institute of Computing Technology, Chinese Academy of Sciences, and University of Chinese Academy of Sciences. Z. Li and Q. Wu are also with the Purple Mountain Laboratories. E-mail: {zyli, lijinyang, wuqinghua}@ict.ac.cn (Corresponding author: Zhenyu Li)*
- *G. Tyson is with the Hong Kong University of Science & Technology (GZ) and Queen Mary University of London. E-mail: gtyson@ust.hk*
- *G. Xie is with the Computer Network Information Center, Chinese Academy of Sciences. E-mail: xie@cnic.cn*

A preliminary shorter version [23] of this paper appeared at IEEE INFOCOM, 2022.

QoE. Finally, we characterize the viewership of individual broadcasters with an aim to find out the loyal viewer. To the best of our knowledge, this work is among the first to examine and optimize MLS systems from both the first-mile and the last-mile aspects. Our measurements reveal four key observations (§3):

- ***Wasteful Uploads:*** We identify significant volumes of redundant live video uploads, which receive no viewers (3.3% of total upstream traffic). Waiting for the first viewer's arrival and viewer clear-outs during streaming are the two dominant contributors. These two reasons constitute 30% of total upstream traffic.
- ***Prevalent Clear-Outs:*** 99.9% of broadcasts experience periods with zero viewers, which we refer to as clear-outs. The clear-out duration is related to broadcaster popularity. Excessive clear-out periods contribute the greatest amount of wastage (23% of total upstream traffic). Moreover, we show that streams are unlikely to become popular when encountering clear-outs in their late stages.
- ***Viewing Locality:*** Even though notionally these platforms target a global audience, streams of individual broadcasters tend to attract viewers from a small set of network regions, indicating a strong viewer locality. The top regions for individual broadcasters vary greatly. We show that this locality impacts video performance. For instance, cross-region delivery of content to viewers doubles the startup delay, compared with same-region delivery.
- ***Loyal Viewers:*** The "loyal" viewers, who regularly watch the same broadcasters, are small in number but generate 59% of the total video data downloaded. These loyal viewers not only arrive earlier during broadcasts but also leave later.

Motivated by these observations, we design EDGEOPT, a centralized control center for MLS services. EDGEOPT optimizes both the first-mile (broadcasters $\rightarrow$ ingest servers) and the last-mile (CDN edge servers $\rightarrow$ viewers) transmission in MLS. For the first-mile transmission, EDGEOPT uses a decision tree model to predict the attractiveness of individual broadcasts when clear-outs happen. It then suppresses the video transmission during clear-outs for uploads, which it predicts will get zero views. As a result, resource wastage is reduced by 71%. EDGEOPT exploits these bandwidth savings to then enhance the quality of highlight segments for remaining time-shifted views. This scheme improves the median quality of the retransmitted video segments by 10%, while introducing only negligible overheads.

For the last-mile transmission, EDGEOPT uses an edge server pre-fetching strategy, composed of a pre-fetching location selection scheme based (on our observations of viewing locality). It also employs a pre-fetching time prediction scheme using a deep neural network. Through this, we reduce startup delay by 29.5%. EDGEOPT also includes a peer-assisted delivery scheme that leverages the upload capacity of loyal viewers to offload traffic from the edge servers. This scheme can reduce the server load by as much as 80%.

In the rest of the paper, we provide background about MLS and introduce our dataset in §2. In §3, we characterize the user behavioral patterns and provide insights for system optimization. In §4, we present the design of EDGEOPT, followed by the evaluation in §5. We survey related work in §6 and conclude the paper in §7.

2 Background & Dataset

2.1 Overview of MLS and Motivation

Overview. We rely on logs shared by a large-scale Chinese Mobile Live Streaming (MLS) platform.[1] This MLS provider offers the same functionality as popular platforms in Western countries *e.g.,* Periscope [8] and Facebook Live [5]. The platform serves millions of users per day and anyone can be a broadcaster, streaming their camera feed. When users start broadcasting, they upload video segments to one of the ingest servers, which in turn publishes the video chunks to a Content Delivery Network (CDN). The CDN is then responsible for disseminating chunks to viewers who request them via HTTP(s). The videos are also (optionally) available after the broadcast, offering users time-shifted "replay" views. MLS platforms often allow viewers to interact with broadcasters, typically via text that is synchronized with a different connection other than the one used for streaming delivery. We note that some MLS platforms (*e.g.,* Taobao Live) also allow broadcasters to invite other users for co-streaming [22], which mixes the two video sources and distributes the mixed stream to viewers.

Motivation. In contrast to traditional live streaming services, where broadcasts are often from professional users (*e.g.,* TV program makers), MLS services allow any user to be a broadcaster. That said, both amateur and professional users are often present. With many broadcasts (especially from amateur users) available, individual broadcasters often find it hard to gain a large audience. Indeed, as we will show in §3, a large amount of content (33.3% of total upstream traffic) that is uploaded does not obtain any viewers. As such, service providers strive to mitigate the wasted uploads while keeping user-perceived quality unchanged. The mitigation, however, requires a deep understanding of user behavior.

Another unique feature of MLS services is that both broadcasts and views are much shorter than traditional live streaming [41]. Hence, viewers are less patient in waiting for the start of playback. While reducing the startup delay through pre-fetching and caching is promising, this depends on several key decisions, such as where to cache and what to prefetch.

Finally, while a low video bitrate is helpful for reducing the latency from broadcasts to their live viewers, it reduces the video quality for time-shifted "replay" views (who are less delay-sensitive due to buffers). We thus need to consider the quality of both live streaming and later time-shifted "replay" views when allocating uplink bandwidth as in [36]. To do so, we also need a detailed analysis of user behavior and a solution to decide the bandwidth allocation.

1. Due to a non-disclosure agreement, we do not share the name.

Motivated by the unique features and requirements of MLS services, we present a detailed analysis of user behavior in a large-scale MLS service and propose EDGEOPT, a system that optimizes both the first-mile (broadcasters → ingest servers) and the last-mile (CDN edge servers → viewers) transmission.

2.2 Dataset Description

We rely on over 1.8TB of logs, covering 10 days of anonymous access shared by a major anonymous MLS service. The dataset covers *all* broadcasters and viewers within the examined period. Within the logs, one live broadcast corresponds to a unique *broadcast ID*. Viewing logs with the same broadcast ID can therefore be connected.

The service logs consist of 1.9M live streams by 0.4M broadcasters; this covers 98 years' worth of video content. For viewers, we have logs including 300M views by 2M viewers, with an aggregated streaming period of 4,394 years. Within each log entry, the major data fields cover four categories:

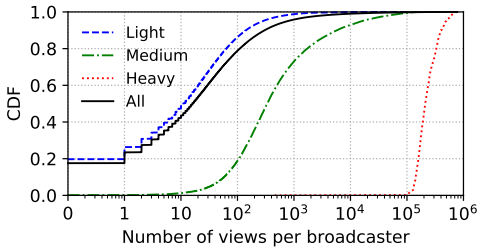
1) *User-specific:* Anonymized user (broadcast ID, resp.), which uniquely identifies a user (broadcast, resp.); anonymized client/server IP,[2] BGP-Prefix, ASN, and province-level geographical location.
2) *Session-specific:* Data volume, duration of the session, direction (upload or download), download traffic rate, and video chunk bitrate.
3) *Viewer-side QoE:* Various QoE metrics, including startup delay, number of buffering events, buffering duration, and the number of retries before successful playback or user giving up.
4) *Operating environments:* Network connection type (*e.g.*, 4G or WiFi), platform type (*e.g.*, Android or iOS).

Although the examined MLS also supports desktop access, 99+% of both the broadcasting and viewing traffic contained in our dataset is from mobile devices.

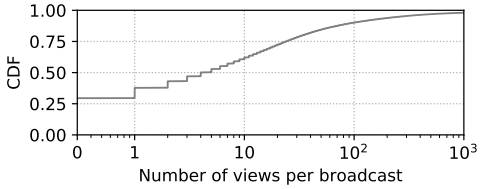
Limitations: As in other measurement-driven studies [28], [35], [41], our dataset covers only one MLS service provider. Given the huge number of sessions contained in the dataset and that the examined service provides similar functionalities as in other popular services (*e.g.*, Facebook Live), the observations made in this paper as well as the optimizations are likely to be applicable to others. Nevertheless, to facilitate research in this area, we make our dataset publicly available. [3] We also note that the 10-day duration of the dataset prevents us from drawing longitudinal conclusions. We, therefore, focus on session-level analysis that does not require long periods of observation.

Ethical Considerations: We took a number of steps to ensure the ethical use of data. We have no access to the content of broadcasts, and can only observe metadata (*e.g.*, session duration). The logs are routinely gathered for operational purposes, and no extra data collection was triggered. All the user information, including user ID, IP address, ASN, and even the broadcast ID, is anonymized. We are unable, and not allowed, to link logs to users.

2. The anonymization is done using `Crypto-PAn` [4].
3. https://www.dropbox.com/s/nmwh75syvi6duxv/MLS-DATA.7z?dl=0

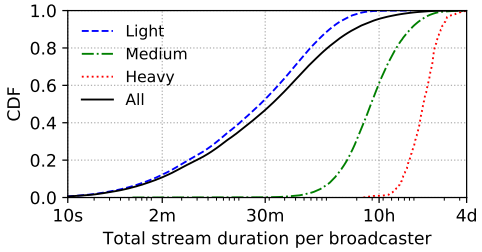


(a) Sum number of views per broadcaster

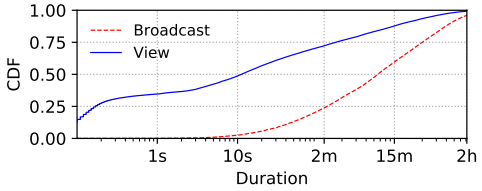


(b) Number of views per broadcast

Fig. 1. The number of views received per broadcaster and broadcast. We later define *Light*, *Medium*, and *Heavy* broadcaster groups (Table 2). The x-axes are on a logarithmic scale.



(a) Sum stream duration per broadcaster



(b) Duration per broadcast

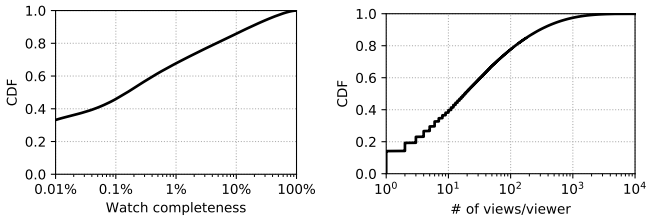
Fig. 2. Duration (*i.e.*, broadcasting length) per broadcaster and broadcast. The x-axes are on a logarithmic scale.

3 MLS User Behavior in the Wild

We start by analyzing user behavior for both broadcasters and viewers. Specifically, we present an overview of user behavior, followed by a detailed analysis of video segments that go unwatched. This later informs our optimizations for the first-mile transmission. We then proceed to analyze geographical popularity and viewer loyalty. This guides the later design of optimizations for the last-mile transmission.

3.1 Characterizing User Behavior

Overall Popularity: We first present the overall statistics of the platform. Figure 1(a) presents the distribution of views per *broadcaster* during the examined 10 days, and Figure 1(b) depicts the number of views per *broadcast*. Although an elite group of broadcasters do gain hundreds of thousands of views, we find that a significant fraction never exceeds 10, and almost a quarter never have *any* viewers.



(a) Watch comp. per view (b) # of views per viewer

Fig. 3. Distributions of watching completeness for each view, and the number of views for every viewer. The x-axes are on a logarithmic scale.

TABLE 1
Viewing statistics for different connection types. *Mix* means connection type switches within the viewing.

	# view	Data volume	Avg. view length
WiFi	80.3%	71.5%	6.06
Cellular	7.4%	4.1%	0.42
Mix	11.5%	23.7%	140.24
Others	0.8%	0.7%	2.15

Despite this, we notice that the App on the broadcaster side keeps uploading content regardless, thereby *wasting* resources without anybody viewing the content.

We next align this with the duration of streams. Figure 2(a) presents the duration of content broadcasted per broadcaster, whereas Figure 2(b) presents the duration of individual streams. Unsurprisingly, the platform is dominated by short-form content. The majority of broadcasts last below 10 minutes, whereas viewing times are *significantly* shorter (median ≈10s). This observation is consistent with Periscope [8] and Facebook Live [5], but contrasts sharply with e-sports platforms like Twitch [10], [16], [47] which have a longer broadcast duration (median exceeds 100 minutes).

Impact of Connection type. We further breakdown viewing statistics by the network connection type, and present the results in Table 1. We observe a WiFi-dominated system, where WiFi carries 71.5% of the download data, and 80.3% of the total viewership. Interestingly, we observe that viewers regularly switch their connection types in the middle of the viewing (*e.g.,* from 4G to WiFi). This occurs for 11.5% of views. In 53% of the connection-switching views, the connection types change within cellular genres (*e.g.,* from 3G to 4G), indicating that the viewer moves to a different network environment. In a quarter of the cases, WiFi connections are replaced by cellular connections, while the opposite switch type (*i.e.,* from cellular to WiFi) happens in 22% of cases. The latter two switch types potentially indicate a bad initial connection, since most of these switches occur in the initial stage of broadcasts (median first 5%). Noticeably, these viewing sessions with connection-switching are the longest (140s *vs.* an overall average of 10s). This is likely because longer viewing sessions are the only ones that warrant performing switches in the access network. The prevalent connection type switching also shows the importance of improved user mobility support through techniques such as QUIC or MPQUIC [51].

Note, we have also looked into the broadcaster's connection type, and found a similar WiFi-dominated pattern.

Wasted Uploads. Despite the short-form content, we find viewers often quit viewing before the end of the broadcasts. To understand this better, we define *watch completeness ratio* as the ratio of the view duration over the broadcast duration. A value of 1 indicates that an entire stream is viewed. Figure 3(a) presents the watch completeness ratio per view. We see that around 30% of views have a completeness ratio of under 0.01%. The median watch completeness is around 0.1%, and only 10% of streams have over 10% of their bytes consumed. These results show further evidence that a significant fraction of *unwatched* content is needlessly uploaded from broadcasters.

To get a handle on the low completeness ratio, Figure 3(b) presents the number of streams viewed by each viewer. We find most viewers watch a large number of distinct videos. The median view count per user is 18, and 22% of viewers watch at least 100 live broadcasts. Thus, we conjecture that many users skip between videos looking for content of interest. This skipping means that users only watch a small fraction of a stream's content. We find that 76% of intervals between two consecutive views are under 1 second, indicating this is indeed the case. To confirm this, we adopt the approach in [19] to separate per-user request sequences into individual sessions. Specifically, we fit a 2-component Gaussian Mixture Model (GMM) to determine the maximum interval between two consecutive views in one session, which is 43.5s. Using this threshold, we then separate requests into individual sessions. As a result, 50% of sessions contain at least 5 video requests, with a maximum of 2,819 and an average of 13 requests. 10% of sessions exceed 33 video requests, revealing a set of highly active users who skip extensively.

The above results reveal that a significant volume of content is uploaded to the servers, yet never consumed. In total 33.3% of upstream traffic is wasted. This is attributed to both zero-viewed broadcasts and the low watch completeness ratio. We make a breakdown analysis of the unwatched content in §3.2.

Broadcaster Clustering: Due to the diversity of broadcast behaviors observed above, we next cluster broadcasters into groups to better understand their patterns. To this end, we choose 5 features: (*i*) the number of live broadcasts (Feature 1, F1), (*ii*) the total broadcasting duration (F2), (*iii*) the number of active days (F3), (*iv*) the total view count (F4), and (*v*) the total view duration (F5). We first use Z-Score [9] and Principal Component Analysis (PCA) [7] to preprocess the data, on which we apply K-Means [6], experimenting with K from 2 to 10. We select K=3 due to a relatively small Davies-Bouldin Index [13].

Table 2 presents the results. The broadcasters can be divided into three main categories: *Light* (L), *Medium* (M), and *Heavy* (H). We thus return to Figure 1(a), which segmented results based on these three groups. The least active/popular broadcasters (Light) account for the vast majority ($\sim$ 90%). We speculate that this type of broadcaster is new to the system, or just experimenting with the live function in the app. The fraction of medium active broadcasters is relatively small ($\sim$ 10%), but their level of activity and popularity is far more than the Light cluster. Such broadcasters have formed a live broadcast habit, and may

TABLE 2
Clustering results for broadcasters (the statistics are shown in median).

Label	%	F1	F2	F3	F4	F5
Light	89.0	2	0.4	1	11	0.1
Medium	10.8	15	8	6	336	14
Heavy	0.2	18	31	10	210,423	29,841

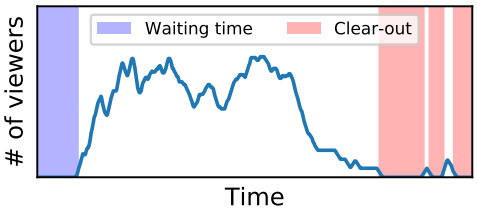


Fig. 4. An illustration of *waiting time* and *clear-out*.

broadcast every day. The most active/popular broadcasters (Heavy) are distinct from the other two types though: These are the *"stars"* of the platform. Although the fraction of these broadcasters is small (0.2%), they contribute a disproportionately large amount of upstream and downstream traffic.

3.2 Characterizing Unwatched Segments

We already know that 33.3% of upstream traffic is wasted (*i.e.,* never viewed). However, despite that 29% of live broadcasts are never watched overall, only 3.3% of upstream traffic is wasted due to this. Instead, the majority of waste comes from two other kinds of unwatched broadcast *segments* (*i.e.,* partially unwatched streams), which we examine for the *first* time: (*i*) 7% of upstream traffic is wasted while waiting for the first viewer to arrive (termed the *waiting time*); and (*ii*) 23% of upstream traffic is wasted because all viewers exit before the broadcast ends (termed a *clear-out*). For illustration, Figure 4 presents an example stream from our dataset. The purple segment (waiting time) highlights the initial 10% of the stream with no viewers, while the three red clips (clear-outs) show where all viewers have exited.

Waiting for the First Viewer: Overall, 7% of upstream traffic is wasted because contents are uploaded before the first viewer arrives. Figure 5 presents the waiting time distribution for broadcaster groups. We find that, in most Heavy broadcasters' streams, the first viewer arrives within 10s, while for broadcasters in the Light and Medium group, most of their streams must wait for $\sim$ 1 minute.

Clear-Out Period: A *clear-out* refers to when all viewers cease consuming a stream, either permanently or temporarily. 23% of total upstream traffic is wasted due to clear-outs (*i.e.,* an upload continuing even if all viewers have left). Clear-outs occur across the whole spectrum of broadcasters: 99% of broadcasts (that have been watched at least once) experience clear-outs, whose lengths range from milliseconds to hours. Figure 6 plots the distribution of the clear-out duration for each broadcaster group. Although the median clear-out duration for Light and Medium broadcasters is around 40s, about 60% of clear-outs for Heavy broadcasters never exceed 1 second, indicating that the clear-out length is related to popularity. In addition, we note that clear-outs are likely to occur more than once per broadcast (4 times median).

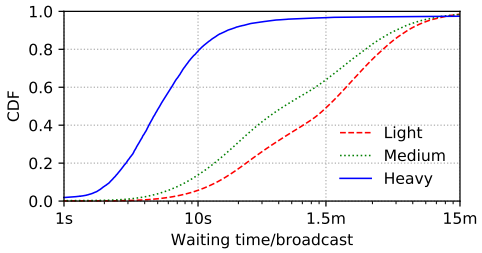


Fig. 5. Waiting time distribution for each type of broadcaster.

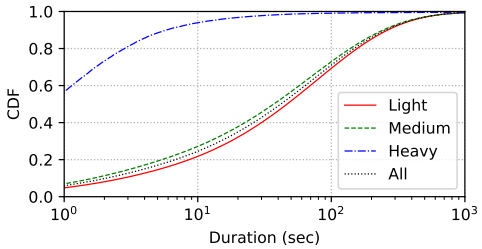
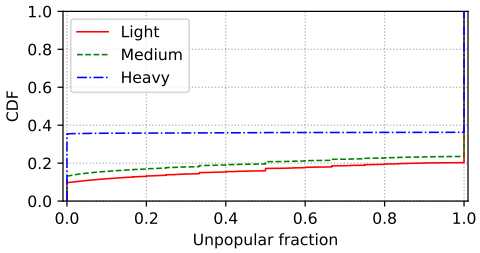


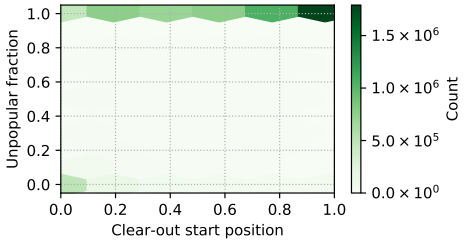
Fig. 6. Clear-out duration distribution for each type of broadcaster.

We also find in 50% of all cases, the interval between clear-outs is no more than 3 seconds, suggesting that a clear-out is likely to be closely followed by another one. This observation indicates that, once a broadcast has started to lose attraction (*i.e.,* experiencing several clear-outs), it is unlikely to regain it. To confirm this, we split the broadcast between the starting points of clear-out$_i$ and clear-out$_{i+1}$ into B 30-sec bins, and count the maximum number of online viewers (v) for each bin. An example of this is shown in Figure 8. We then record the run length (R) of unpopular bins (whose $v \leq 1$). Finally, we define the *unpopular fraction*, f, as $f = \frac{R}{B}$. For example, in Figure 8, $B = 3$, $R = 2$, and thus $f = 2/3$. The larger the value of f, the less popular the segments between the two clear-outs are. We present the distribution of f in Figure 7(a) over all clear-outs for the three types of broadcasters, where we observe a bi-modal distribution: for Light/Medium broadcasts, in most cases (75+%), a clear-out means that the following content is not popular ($f = 1$). For Heavy broadcasts, in 40% of cases, even if a clear-out happens, it gains more than one viewer within 30 seconds ($f = 0$).

Next, we examine the correlation between the unpopular fraction, f, of a clear-out and its (normalized) start position, where we represent the start position as the time offset (relative to the stream's start time) where the clear-out happens normalized by the stream duration. We plot the results as a heatmap in Figure 7(b), where a darker color implies more clear-outs are within this area. We find that the clear-outs with f=0 mostly occur at the initial stage of the broadcasts: for $L/M/H$ live broadcasts, they occur at 0.11, 0.03, and 0.003 (median) of the broadcast lifetime, respectively. These early clear-outs are probably due to the small number of viewers in the early stage, and clear-outs will thus accidentally occur. In contrast, those clear-outs with f=1 usually happen in the late stages of the broadcasts (their median start positions are 0.65, 0.72, and 0.99 for the $L/M/H$ broadcast, respectively). That said, the clear-outs occurring in the late stage of broadcasts indicate the



(a) Unpopular fraction distribution



(b) Unpopular fraction *vs.* clear-out start position

Fig. 7. The unpopular fraction is related to the broadcaster group and where the clear-out happens.

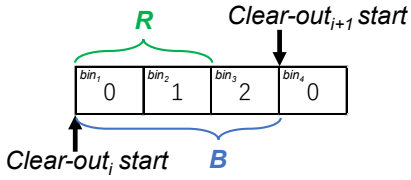


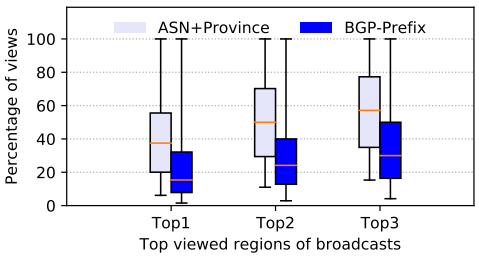
Fig. 8. An illustration of unpopular fraction calculation.

content is losing attraction, and thus can be suppressed to save traffic. We later confirm this implication in §4.2, using a decision tree model.

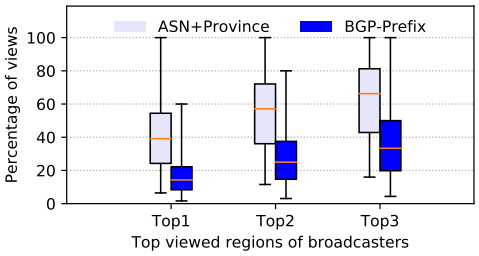
3.3 Geographical Popularity

Defining Locality: We next investigate the geographical popularity of broadcasters with a focus on whether the viewership of individual broadcasters is concentrated on a handful of networks. This is important if edge caches or peer-assisted delivery were to be deployed. Since we are interested in users' network footprint, we use the BGP-Prefix and ASN+Province of the user's IP address to represent the location. Note that we use the ASN+Province combination, as a given ASN may have a presence in multiple provinces [40].

Demand Aggregation: To explore the potential of network-level demand aggregation, we compute the percentage of views that come from the top k regions, where $k \in \{1, 2, 3\}$. We exclude the Light broadcasters to reduce the randomness introduced by inactive users. Figure 9 presents the proportion of views that fall into the top three regions on a per-broadcast and per-broadcaster basis. We see a large fraction of broadcasts attract views from just a few locations. About half of live broadcasts receive over 50% of their total viewership from just 3 ASN+Provinces. This percentage is 30% when considering each broadcast's top 3 BGP-Prefixes. This suggests strong network localized viewing patterns.



(a) Broadcast



(b) Broadcaster

Fig. 9. Percentage of views contributed by the top three regions.

The above observation is also mirrored from the broadcasters' point of view, with an even larger (+5%) proportion of views generated from their top locales. Moreover, the concentration of views can be observed at a global level: the top 50 (2%) ASN+Provinces or top 500 (5%) BGP-Prefixes contain 80% of all viewers. This later inspires us to propose a pre-fetching scheme to better localize content (§4.4).

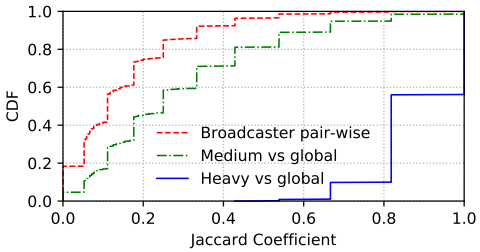
We also examine whether the top regions of individual broadcasters differ from each other, or if they overlap with the global top regions (*i.e.*, the regions contributing the most traffic globally). To this end, we extract for each broadcaster b the top 10 regions contributing the most downstream traffic from b's streams (r_b for short). We then calculate the *Jaccard Coefficient* [21] for every pair of broadcasters (r_i, r_j). A higher coefficient implies more overlap between the top regions of the two broadcasters. Figure 10 plots the distribution of the coefficient.[4] We observe a small median Jaccard coefficient (0.11 at ASN+Province level and 0 at BGP-Prefix level), indicating broadcaster-specific locality.

We further report for each (Medium or Heavy) broadcaster, the Jaccard coefficient between the set of their top 10 regions and the set of top 10 regions globally (blue/green lines in Figure 10). The global top regions are largely defined by Heavy broadcasters, with a median coefficient of 0.81 at ASN+Province level, and 0.66 at BGP-Prefix level. In contrast, the ones from the Medium group deviate from the globally popular locations significantly, indicating the need for per-broadcaster predictions for techniques like pre-fetching (§4.4).

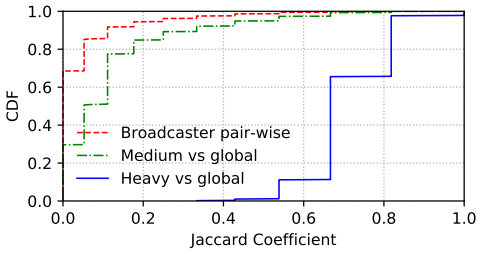
Impact on QoE: We observe that, despite strong viewer locality, the majority of streams (from servers to viewers) cross network boundaries: 88.6% of the views transmit data across ASN+Province boundaries, whereas 99.8% of broadcasts stream to a different BGP-Prefix.

The above indicates significant scope for localizing traffic via techniques such as pre-fetching and peer-assisted deliv-

4. We excluded Light broadcasters to prevent randomness.

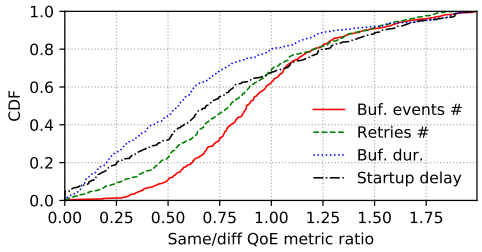


(a) ASN+Province

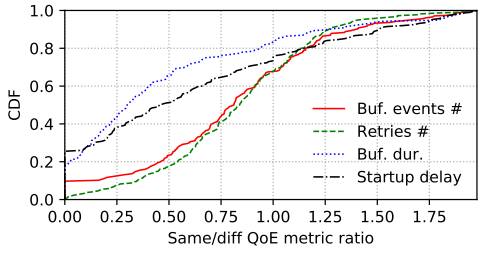


(b) BGP-Prefix

Fig. 10. Overall small Jaccard coefficient indicates broadcaster-specific locality.



(a) ASN+Province



(b) BGP-Prefix

Fig. 11. Distribution of $\frac{same\ region}{cross\ region}$ QoE metrics in ASN+Province networks (a) and BGP-prefix networks (b).

ery. To understand the potential QoE benefits, we calculate four relevant viewer metrics[5]: (*i*) startup delay, (*ii*) number of buffering events, (*iii*) buffering duration, and (*iv*) number of connection retries before success or abandonment. We calculate these metrics for each network region (ASN+Province or BGP-Prefix) that has a CDN server receiving over 10 views. We separate views into those that come from the same region and those that come externally. Finally, we calculate the ratio of the two averages for each metric as $\overline{v}_{same}/\overline{v}_{cross}$.

Figure 11 presents the distribution of ratios across all considered network regions. In most cases, the ratio is less than 1, which means the QoE metrics are better for same-region delivery. Specifically, in half of the cases, cross-region delivery doubles the startup delay and buffering duration (the most important two metrics for video streaming), compared with same-region delivery. In addition, the ratio is smaller in the same BGP-Prefix delivery than in the same ASN+Province case, which is expected as BGP-Prefix is a smaller network region. This confirms that bringing content closer to consumers (*e.g.*, via pre-fetching) could significantly improve QoE.

3.4 Loyal Viewers

We finally inspect "loyal viewers", who regularly view individual broadcasters. We posit such viewers may be highly predictable and therefore suitable for optimization via predictive content pre-fetching. To the best of our knowledge, we are the first to explore loyal viewers in this domain, without reliance on explicit information (*e.g.*, follower list [28]).

5. We do not consider the live streaming delay (*i.e.*, the time elapsed from the live event occurring at the broadcaster to being displayed at the viewer) because it depends on several factors, including the encoding time at the broadcaster side, the first-/last-mile delay, the CDN delay, as well as the buffering and decoding time at viewer side. This paper focuses on the first-/last-mile delay. Interested readers are referred to [22] for details of reducing CDN delay.

TABLE 3
Clustering results to find loyal viewers (the statistics are shown in median).

Label	%	Ratio of #	Ratio of dur.
Normal viewers	85	6.25%	0.01%
Borderline loyal viewers	13	18.18%	0.94%
Core loyal viewers	2	27.78%	12.41%

Identifying Loyal Viewers: To determine whether viewer v, who has watched broadcaster b's streams, is a loyal viewer of b, for each pair of (v, b) we extract 2 features: (*i*) the ratio of view counts to broadcast number; and (*ii*) the ratio of the total viewing time to the total live broadcasts' duration. To extract broadcaster and loyal viewer relationships, we perform clustering for pairs of (v, b) on the above 2 features using K-means. We experiment with K from 2 to 10, and set K to 3, due to the relatively small DBI.

The clustering results are shown in Table 3. The viewer/broadcaster pairs are divided into 3 main sub-populations, with the least loyal group (normal viewers) accounting for the largest proportion (85%). In contrast, the level of loyalty within the other 2 groups (borderline/core loyal viewers) is stronger. For the core group, the viewers watch more than a quarter of a broadcaster's streams, and the viewing duration is higher than the other groups. We will refer to the two groups of viewers, except normal viewers, as loyal viewers hereafter.

Characterizing Loyal Viewers: We next inspect loyal viewers' characteristics across the entire dataset. In a live broadcast, on average, loyal viewers account for only 12% of the total viewers. Yet they generate the majority of download traffic: the median percentage of download volume contributed by loyal viewers per stream is 18% and 55% for Medium and Heavy broadcasters, respectively. In total, 59% of video data is downloaded by loyal viewers, which suggests that optimizing for this small fraction of users would be disproportionately beneficial.

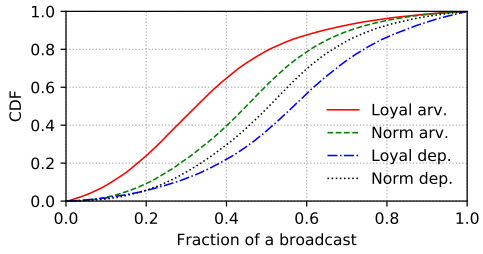


Fig. 12. The position of a broadcast where loyal/normal viewers arrive/depart.

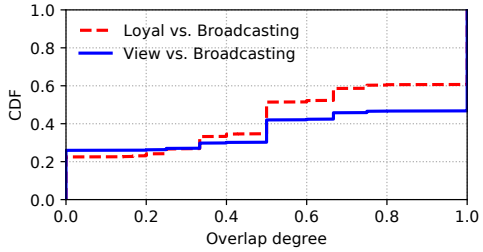


Fig. 13. Overlap degree of broadcasters' and viewers' locations. We separate viewers into loyal viewers and all.

Another interesting observation is that, compared with normal viewers, loyal viewers' viewing is more proactive in terms of their arrivals and departures. To quantify this, we inspect the position where loyal/normal viewers arrive/depart, and plot the results in Figure 12. Loyal viewers start watching earlier than normal users (median position 0.32 *vs.* 0.44), and leave later (median position 0.56 *vs.* 0.50). Furthermore, the viewing sessions of loyal viewers are significantly longer than that of normal ones (432s *vs.* 11s, on average). The above confirms that, unlike normal viewers who tend to frequently switch broadcasts (§3.1), loyal viewers consume persistently. As such, we posit that loyal viewers are potentially good candidates for stable peers in peer-assisted delivery [26].

To get a deeper understanding of these loyal viewers, we investigate whether loyal viewers share the same locality (ASN+Province) as the broadcaster. To this end, we define Overlap Degree (O) as:

$$O(locs_{brd}, locs_{loy}) = \frac{locs_{brd} \cap locs_{loy}}{min(|locs_{brd}|, |locs_{loy}|)} \quad (1)$$

where $locs_{brd}$ denotes the set of top 10 regions where the broadcaster broadcasts. $locs_{loy}$ indicates the set of top 10 regions where the loyal viewers generate the most views. The higher the O is, the more similar the two sets of regions are (*i.e.,* the more overlap). The resulting distribution of O for the broadcasters who have loyal viewers is presented in Figure 13. We observe that the median overlap degree is 0.5. This suggests that half of the loyal viewer population is within the same region as the broadcaster. Furthermore, for as many as 40% of broadcasters, their O value equals 1. The high overlap is probably because they share the same locality or dialect. We further prove this by inspecting the overlap degree for broadcasters *vs.* all their viewers (see Figure 13). We again see a higher overlap degree, where O values of more than 50% of broadcasters equal 1. The above results highlight the value of location-based broadcast recommendations in MLS.

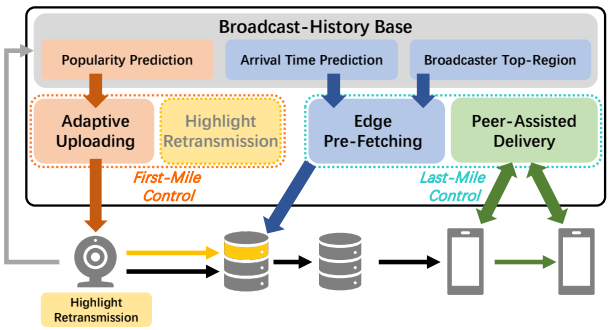


Fig. 14. The system structure of EDGEOPT. The camera indicates the broadcaster; the phones indicate consumers. The upper block is centrally hosted. The servers (bottom) are distributed at the edge. Blocks and arrows are color-coded to indicate components that contribute to the same optimization.

3.5 Takehome Messages

Redundant Uploads: A significant fraction of video content is uploaded by broadcasters but never consumed, resulting in 33.3% of total upstream traffic being wasted. About 30% of live broadcasts go entirely unwatched, but this only makes up the *minority* of the wastage. The dominant contributors are partially unwatched broadcasts (*i.e.,* the waiting time and clear-outs). Thus, suppressing redundant uploads of unwatched content could mitigate the traffic load (§4.2).

Excessive Clear-Outs: Clear-outs cause the greatest amount of wasted upload traffic. Clear-outs are commonplace (99% of broadcasts experience clear-outs) and may last hours. In particular, clear-outs that happen in the later stages of a live stream are a good indicator that no future viewers will arrive. We conjecture that the saved bandwidth could then be leveraged to enhance the quality of the later time-shifted "replay" views (§4.3).

Predictable Locality Traits: Broadcasts show strong localized viewing patterns. Over half of broadcasts receive >50% (resp. 30%) of their viewership from their top 3 ASN+Provinces (resp. BGP-Prefixes). Notably, the top regions vary significantly among (medium active) broadcasters. In addition, transferring content from the CDN servers to the viewers across network boundaries doubles the startup delay and buffering duration in 50% of cases. Techniques such as end server pre-fetching to localize the delivery may therefore improve viewers' QoE (§4.4).

Loyal Viewers: Some popular broadcasters are viewed regularly by loyal viewers. Although the number of loyal viewers is small, they consume the majority of download traffic (59%). Further, loyal viewers arrive at a broadcast earlier and also leave later than others. This makes loyal viewers easy to predict for pre-fetching and peer-assisted delivery ((§4.5)).

4 EDGEOPT DESIGN

Inspired by our observations, we propose EDGEOPT to save resources (for transmission and storage) and improve

user-perceived video quality. Conceptually, EDGEOPT is a centralized control plane for MLS services with assistance from clients.

4.1 Overview

EDGEOPT introduces a set of four optimizations. These are split between the client and server sides, with both coordinated by a central controller. These optimizations cover both the first-mile and last-mile, each designed with different goals. The *first-mile control* tries to save network and storage resources by suppressing upload wastage via adaptive uploading (at the client-side). We then reallocate the saved bandwidth to enhance the quality of time-shifted video encoding. The *last-mile control* aims to improve the viewing experience with prediction-based content pre-fetching (at the server-side). It also strives to mitigate potential CDN "hot spots" through peer-assisted delivery, combining both the clients and servers. All of these optimizations are coordinated centrally via EDGEOPT.

To inform the decision-making, EDGEOPT relies on a *Broadcast-History Base* (BHB) that records historical statistics on a per-broadcaster basis. Note that while logically centralized, each component in EDGEOPT can be replicated in a number of networks to improve scalability and responsiveness. The rest of this section details each optimization in turn.

Figure 14 presents an architectural overview of EDGEOPT. It consists of several modules, which collectively deliver the *four* optimizations. We color-code the figure, to indicate parts that contribute to the same optimization. Adaptive Uploading, Edge Pre-Fetching, and Peer-Assisted Delivery are all coordinated centrally, whereas Highlight Retransmission operates exclusively on the client (hence the dotted line border in Figure 14). The bottom left-hand side camera represents the broadcasters, whereas the phones indicate consumers. The figure shows that all broadcasts are streamed via edge servers, whereas the Peer-Assisted Delivery compliments this with peer-to-peer segment exchange.

4.2 Adaptive Uploading

To mitigate wasted segment uploads when there are no viewers, EDGEOPT employs *Adaptive Uploading* (AU). Specifically, EDGEOPT uses a centrally trained (decision tree) model to predict the "attractiveness" of a broadcast. If the broadcast is predicted to be "unattractive", then the central AU module instructs the broadcaster (client) to upload at a low bitrate (*e.g.*, 144P) until the first viewer arrives, at which time the bitrate will switch to normal. We consider a broadcast unattractive if its *unpopular fraction*, f (defined in §3.2) equals 1. Put simply, an unattractive live broadcast will *not* have more than one viewer simultaneously until the next clear-out occurs, and this is agnostic to the choice of the bin length in f's definition (§3.2).

We choose a decision tree [27] as our classification model due to its efficiency and interpretability. We select 3 features that are easy to measure and understand: (*i*) the length of the broadcast so far (len_b); (*ii*) the number of clear-outs that have previously occurred in the broadcast (n_{clear}); and (*iii*) the maximum f of the broadcast so far (max_f). We then train a model to predict for each clear-out whether the corresponding unpopular fraction f equals 1. The model can be trained periodically to incorporate the latest statistics.

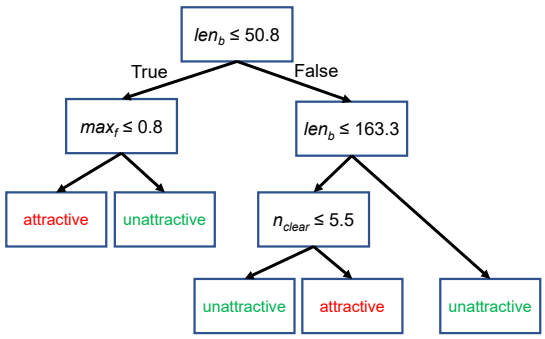


Fig. 15. The decision tree trained for clear-out classification. len_b is the length of the broadcast so far; n_{clear} means the number of clear-outs that have occurred in the broadcast; max_f is the maximum f of the broadcast so far.

We use the first 60% of the clear-out statistics collected from all broadcasts for training and the rest for testing. After training, the depth of the resulted decision tree is 245. We further exploit Cost Complexity Pruning (CCP) [27] to prune the decision tree. The resulting tree structure is presented in Figure 15. It achieves 85% precision, 96% recall, and 0.9 F1 on the testing set. The length of the broadcast so far (len_b) has the biggest impact on the results. This confirms our previous observation that later clear-outs in a stream are a sign of losing attraction (§3.2).

We emphasize that, if the broadcast is predicted to be unattractive, the video keeps uploading but at a lower bitrate. As such, an incorrect prediction only affects the first viewer in that they may receive a lower bitrate than expected. This impact, however, only lasts for a short time period because the video will soon switch to a normal bitrate.

4.3 Highlights Retransmission

Besides live streaming, MLS services often allow time-shifted viewing, where the uploaded live contents are archived for later replay. However, to keep latency low during live streaming, broadcasts are usually encoded at a low quality (*i.e.*, 240P/360P). To enhance the quality of the video segments that are likely to be replayed, we propose a *highlight quality-enhancement retransmission* scheme, which leverages spare capacity at the broadcaster side during unwatched periods.

A video is divided into segments of fixed length L (10s by default, the median viewing duration), and encoded in two bitrate versions (normal and high) in parallel like simulcast [12]. The normal quality encoding is uploaded in real time for live viewing, while the high-quality encoded version is cached locally for possible transmission. Our scheme then exploits the spare bandwidth in the clear-outs during which live uploading is suppressed to a low bitrate (144P in our design). During these periods, the client transmits the high-quality versions of the segments that are likely to be replayed later.

The key challenge is to decide which video segments should be retransmitted. This priority is calculated locally by each broadcaster (as indicated by the yellow box below

the broadcaster's camera icon in Figure 14). Based on the examined MLS service team's experience, time-shifted viewers are mainly interested in highlight clips (*e.g.,* Teamfights in the League of Legends game), instead of the entire stream. Therefore, we order the cached video segments based on the number of viewers (v) as well as the video quality (Q) in the live streaming. Specifically, we assign each cached segment i (high-quality versions) a score S_i:

$$S_i = \frac{v_i}{Q_i},\ Q_i = 1 - \frac{1}{(2b_i + 1)} \tag{2}$$

where Q_i represents the SSIM (structural similarity index) of the segment, b_i is the average bitrate of the segment in live streaming, and v_i is the number of viewers of the segment. The higher the score is, the higher the priority is for the segment to be retransmitted.

Note that our approach retransmits the high-quality versions for at most n (we set n to 10) segments, in order to reduce the local storage usage for caching the high-quality versions and the bandwidth for retransmission.

4.4 Edge Server Pre-Fetching

High video startup delays negatively impact viewer QoE [17], [20]. In §3.3 we observed geographically localized viewing patterns, indicating that there is scope to reduce start-up delays by placing content in consumers' locales. Due to this, EDGEOPT integrates an *Edge Server Pre-Fetching* scheme. This preemptively pushes content to edge servers, where segments are predicted to be viewed in a given locale.

Overview: We place local cache servers in each region (ASN+Province or BGP-Prefix) that users share. If a locale is selected for pre-fetching a particular broadcast, its video segments will be continuously pushed to the selected cache server(s) as a stream of Group of Pictures (GoP). Here, a GoP is set to 120 frames [44]. There are three key challenges in the pre-fetching scheme design: (*i*) *what* to pre-fetch; (*ii*) *where* to pre-fetch; and (*iii*) *when* to pre-fetch. We discuss each of these below.

What To Pre-Fetch: EDGEOPT attempts to push the most popular content. To achieve this, it relies on the historical popularity of broadcasters, as this is a good predictor of future popularity. EDGEOPT pre-fetches *all* broadcasts of Medium/Heavy broadcasters as they consistently accumulate large audiences (see Table 2).

Where To Pre-Fetch: For each broadcaster, we select the top k regions from edge server locations where viewers generate the most historical views on their broadcasts. EDGEOPT forwards the selected broadcaster's stream to these k server locations (regardless of client requests). Clients wishing to consume a stream then send their requests via their local cache server. If a local copy exists, it is immediately returned; otherwise, the request is forwarded to the backend (and then cached for subsequent requests). In all cases, only the *latest* GoP is cached, as older ones are not useful for live streaming.

When To Pre-Fetch: Pre-fetching too early (*i.e.,* long before a viewer's arrival) brings no benefits and only wastes bandwidth resources. Thus, our objective is to ensure the predicted arrival time, Arv_{pred}, is close to the actual arrival time, Arv_{actual}. This must reserve sufficient time for the content to be relayed from the origin server to the edge before the viewers' arrival. Formally, Arv_{pred} and Arv_{actual} should satisfy the following constraints:

$$Arv_{actual} - Arv_{pred} - t_{relay} \geq 0 \tag{3}$$

$$Arv_{actual} - Arv_{pred} - t_{relay} \leq \epsilon \tag{4}$$

where ϵ is a very small non-negative number, and t_{relay} is the time consumption of relay transmission. For t_{relay}, we use 200ms – the median relay delay in our dataset.

To predict the viewer arrival time, EDGEOPT relies on a deep neural network based regression model. The architecture is presented in Figure 16. For each broadcast, we gather 64 features in 5 categories (see Table 4) from its broadcaster's past 5-day of activity; these statistics are stored in the BHB. We standardize the feature vectors by using the Robust Scaler [3], to avoid the influence of outliers. The neural network first encodes the six multi-dimensional features

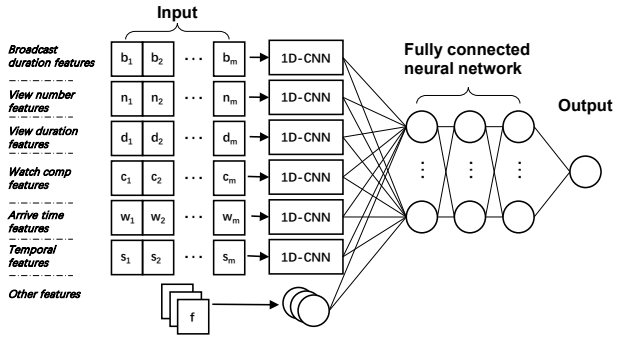


Fig. 16. The DNN model for arrival time prediction.

TABLE 4
Description of features used for arrival time prediction.

Index	Feature name
Cat. #1 Broadcaster-specific	
0	Total number of broadcasts
1	Broadcaster type
[2-9]	Broadcast duration features
10	Number of active days
11	Daily median number of broadcast
Cat. #2 Broadcaster viewing-specific	
[12-19]	View number features
[20-27]	View duration features
[28-35]	View completeness features
Cat. #3 Arrival time-specific	
[36-43]	Historical arrival time features
[44-51]	Broadcasts of same hour features
Cat. #4 Loyal viewer-specific	
52	Number of all unique viewers
53	Number of viewers watched all broadcasts
54	Number of viewers watched $> 50\%$ broadcasts
55	Number of loyal viewers
Cat. #5 Broadcast-specific	
56	Duration of broadcaster's last stream
57	View number of broadcaster's last stream
58	Arrival time of broadcaster's last stream
59	Broadcast start time
60	Time elapsed from the last broadcast
61	Last broadcast is watched or not
62	Last broadcast's number of clear-out
63	Last broadcast's clear-out duration

Algorithm 1: Finding & connecting to a good peer

```
   input : Stream ID: sid, Viewer information: viewer
1  edge, peer ← RequestIngress (viewer, sid);
2  if peer!=NULL then
3  |  connect (viewer, peer, sid);
4  |  IdleConn (viewer, edge, sid);
5  else
6  |  connect (viewer, edge, sid);
7  Function RequestIngress (viewer, sid):
8  |  edge = FindEdge (viewer.NetAddr);
9  |  if SafeUtilization(edge) then
10 |  |  return edge, NULL;
11 |  peer = FindPeer (sid, viewer);
12 |  return edge, peer;
13 End Function
14 Function FindPeer (sid, viewer):
   |  // Sort swarm peers by watching lengths
15 |  swarm ← FindSwarm (sid, viewer.NetAddr);
16 |  loyals ← FindLoyalViewers (swarm, sid);
17 |  for peer ∈ loyals do
18 |  |  if peer.used == NULL and peer.pathLen < 3 then
19 |  |  |  peer.used ← viewer;
20 |  |  |  return peer;
21 |  for peer ∈ swarm − loyals do
22 |  |  if peer.used == NULL and peer.pathLen < 3 then
23 |  |  |  peer.used ← viewer;
24 |  |  |  return peer;
25 |  return NULL;
26 End Function
```

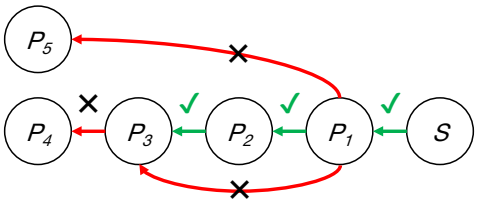


Fig. 17. An illustration of valid and invalid peer connections. S is the edge server, and P_i indicates a peer. The arrows indicate the transmission directions.

(*i.e.*, six 8-tups) into six new features by six 1-Dimensional Convolutional Neural Networks (1D-CNNs). Then, the new features and the remaining features are input into a fully connected network with three hidden layers.

Noticeably, in order to meet constraints 3 and 4, we design the following asymmetric loss function to penalize the overestimation of arrival time, while ensuring that the difference is no less than t_{relay}:

$$loss = diff^2 \times (sign(diff) + \alpha)^2 \quad (5)$$

$$diff = Arv_{pred} + t_{relay} - Arv_{actual} \quad (6)$$

where $sign(.)$ is a sign function, which returns -1 if the input is negative, 1 otherwise. $\alpha \in (0, 1]$ is used to penalize the overestimation. A larger alpha can reduce the chances of overestimation, and we use $\alpha = 0.95$ in our experiments.

4.5 Peer-Assisted Delivery

Finally, EDGEOPT proposes a peer-assisted delivery scheme to alleviate the load on the edge servers that serve viewers. This scheme is promising given the existence of loyal viewers who are more stable during broadcasts. These stable loyal viewers are good candidates for "super peers".

To this end, we equip EDGEOPT with a Peer-Assisted Delivery module (PD), which acts as a centralized control unit for peer-assisted delivery. Algorithm 1 shows the workflow of peer management. Specifically, viewers who watch the same broadcast form a swarm, and all the clients within a swarm are indexed by the central PD module. The stream is divided into Groups of Pictures (GoPs). Upon downloading a GoP, clients of the loyal viewers can share it with other viewers (peers) in the same swarm.

A new viewer first contacts the central PD module to discover which edge server to connect to. The PD in turn locates a nearby edge server and may also return a candidate peer. If the edge is not heavily loaded (*e.g.*, resource utilization $< 50\%$) or there is no available candidate peer, the viewer will be redirected to the edge server. Otherwise, the candidate peer will be designated as the viewer's data source. The PD module tracks each peer's status, and in case of transmission failure, the edge server is used for data transmission. Each peer also maintains a connection with the edge server for backup purpose.

To reduce cross-region traffic, viewers only connect to peers within the same region. Only peers connected via WiFi can serve as sources (to avoid issues with mobile data tariffs). Moreover, in order to prevent peers from being overloaded, we set several constraints on peer selection. These constraints are illustrated in Figure 17: (*i*) A peer can only serve one viewer ($P_1 \rightarrow P_5$ is invalid); (*ii*) One viewer can only have one peer as a data source ($P_1 \rightarrow P_3$ is invalid); and (*iii*) The overlay path length from the peer to the edge server (S in the figure) is at most 3 ($P_3 \rightarrow P_4$ is invalid). Individual clients also report the data transfer performance periodically to the PD module, which will instruct the clients to switch back to the edge server if any performance issues are detected. In the case of connection switching, besides the pre-established backup path mentioned before, the playback buffer at the viewer side will help mitigate stalls.

Finally, service providers can let viewers opt out of peer-assisted delivery. The PD module also records the contributions of each peer, which can be used for incentive purposes. For example, the participants could be rewarded with credit points based on their contribution, where the credit points can be used for in-app purchase (*e.g.*, giving gifts to broadcasters).

5 TRACE-DRIVEN EVALUATION OF EDGEOPT

In this section, we evaluate EDGEOPT, using our trace data (see §2.2).

5.1 Evaluation of Adaptive Uploading

We first evaluate the efficacy of our adaptive upload strategy, which reduces the encoding rate for videos predicted to have zero viewers. In this design, the broadcaster first contacts the Adaptive Uploading module of EDGEOPT to decide whether to reduce the encoding rate. Recall that EDGEOPT centrally makes this decision using the trained decision tree (see Figure 15).

To evaluate this, we replay the broadcast and viewing records from our dataset and calculate the potential savings.

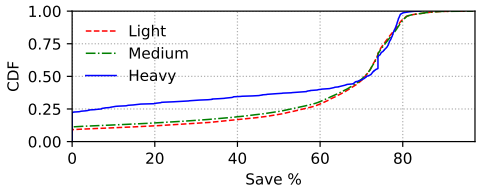


Fig. 18. The percentage of traffic being saved in the clear-out periods for each group of broadcasters.

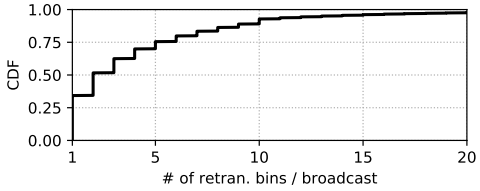


Fig. 19. The number of retransmitted segments per broadcast.

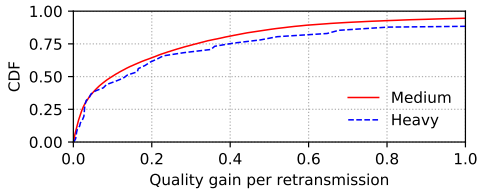


Fig. 20. Video quality improvement by highlight retransmission for two types of broadcasters.

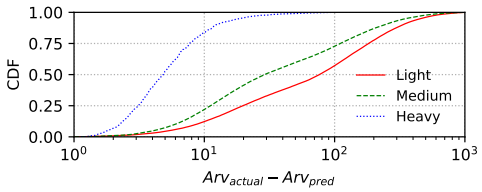


Fig. 21. CDF of $error = Arv_{actual} - Arv_{pred}$ for each broadcaster type. The x-axis is in second.

We confirm that EDGEOPT can save as much as 71% of the data volume during the clear-out period; this accounts for 16.3% of the total uploading traffic. Figure 18 plots, for each type of broadcaster, the traffic savings. The median saved amount for all three types of broadcasters is ∼70%, showing that the adaptive upload scheme can substantially mitigate the wastage issue. We also notice that only a few broadcasters can save beyond 80%. This is because the encoding rate is only reduced, rather than being ceased. In addition, we see that a higher percentage of broadcasters (22%) in the Heavy group receive zero benefit (when compared to the other two types, 10%). This is expected as heavy broadcasters are more likely to be predicted as "attractive" during the whole course of live streaming.

5.2 Evaluation of Highlight Quality Enhancement

To evaluate the benefits of the highlights retransmission scheme, we replay the service logs of Medium (M) and Heavy (H) broadcasters (Light ones are excluded, since they are inactive). Note that in our experiments, we do not estimate the available bandwidth (as the first step of our scheme), since it has been broadly studied (*e.g.*, GCC [14]) and is beyond the scope of this paper. Instead, we rely on the real bandwidth values recorded in the service logs. After the trace-driven experiment, we find that, 46% of the M/H broadcasters' streams have at least one highlight segment being retransmitted. We further examine for each broadcast the number of retransmitted segments and present the results in Figure 19. On average, 4.4 segments (∼44s video) are uploaded for replay enhancement, which is beneficial as our goal is to enhance only the highlights.

Next, we investigate the video quality gain brought about by the retransmission. To this end, for each retransmitted segment, we compute its quality gain:

$$gain = \frac{q(b_{new}) - q(b_{old})}{q(b_{old})} \quad (7)$$

where $q(b_{new})$ is the bitrate of the retransmitted high-quality version, while $q(b_{old})$ is the original bitrate in live streaming. Figure 20 reports the distribution of quality improvement for individual retransmitted segments, where we separate the results according to the broadcaster type (Medium/Heavy). We can observe that the distributions for Medium and Heavy broadcasters are quite similar – the median quality gain is 10%, and about a quarter can achieve 30% quality gain.

Overhead Analysis: A broadcaster needs to cache the high-quality versions of N segments locally for possible retransmission. As at most 10 segments of a broadcast are retransmitted, we can only keep the top 10 segments according to their scores (Eq. 2). In our implementation, we set the high-quality segments to code at 1080P. Assuming standard bitrates [1], the estimated storage cost is 116MB for a broadcast, which is moderate for most mobile devices.

5.3 Evaluation of Edge Server Pre-Fetching

We next evaluate the pre-fetching strategy employed by EDGEOPT.

Prediction Accuracy: First, we use 5-fold validation to evaluate the DNN-based pre-fetching time prediction model. We plot the distribution of prediction errors ($Arv_{actual} - Arv_{pred}$) in Figure 21. We observe that the Heavy broadcasts are most predictable, with a median error of 4 seconds, while the value for Medium ones (resp. Light) is 27s (resp. 73s). We underline that the seemingly high prediction error is unsurprising, since such *pre-publication prediction* is proved to be very difficult [29], [31], due to the uncertainty of viewers' interests and little to no information is available when the broadcast initiates. Further, our intention to avoid overestimating distorts the model to some degree.

Due to this, our focus is not on attaining a very high prediction accuracy, but to use the result as a pre-fetching timing reference. With that in mind, as long as the predicted value is less than the actual value (*i.e.*, satisfying restrictions 3 and 4) fewer wasteful uploads will occur. In this respect, our model can meet the constraints in over 99% of cases. Finally, EDGEOPT helps each broadcast save 10 seconds of uploading content, on average.

Efficacy of Pre-Fetching. We define the metric *view coverage* as the ratio of views localized by pre-fetching, over the

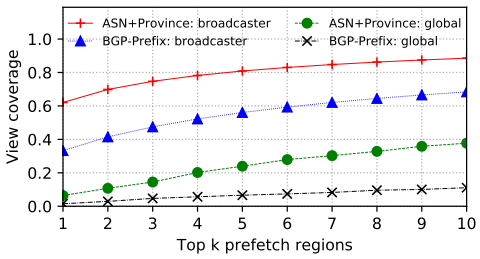


Fig. 22. View coverage of pre-fetch strategies, when pre-fetch at different number of locations.

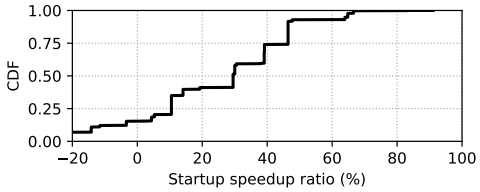


Fig. 23. Startup speedup ratio for each network region.

number of views for individual broadcasts. The closer the view coverage is to 1, the better the efficacy of pre-fetching. We use the first 5 days of viewing records as the training set (to compute the top regions), and the remaining 5 days to test. For comparison, we use a baseline strategy where contents will be pre-fetched to the global top k regions.

Figure 22 presents the results of the average view coverage for all broadcasts when pre-fetching at the k top locations. We see that, to cover half of the views (*i.e.*, view coverage$\geq$0.5), we need to pre-fetch GoPs at only 1 ASN+Province or 4 BGP-Prefixes per broadcast. This is driven by the highly localized viewing patterns (§3.3). While fewer cache servers are needed to achieve the coverage in the ASN+Province configuration, the servers are likely to be further from the viewers than the BGP-Prefix. Note, a larger k naturally leads to better results because more replicas are pre-fetched across regions. But, increasing $k > 5$ only results in marginal improvements. Last, the broadcaster-specific server selection strategy clearly outperforms the baseline (pre-fetch content for global top locations). This is due to the broadcaster-specific viewing patterns (§3.3).

Startup Delay Improvement: Next, we empirically evaluate the startup delay improvement attained by pre-fetching. Again, the content is pre-fetched to the broadcasters' top 5 ASN+Provinces. To model the same-region and cross-region startup delay, we extract (from the dataset) the mean startup delay values for viewers requesting servers of the same-region and different-regions. For each network region that is selected for pre-fetching, G_i, we measure the improvement of startup delay as the *Startup Speedup Ratio*:

$$R_i = 1 - \frac{delay_i^s}{delay_i^c} \tag{8}$$

where $delay_i^s$ is the average startup delay over the views with *both* viewers and servers being located in G_i; and $delay_i^c$ is the average startup delay over the views with viewers being located in G_i *but* servers in different regions other than G_i. We present the full distribution in Figure 23. This shows that the startup delay of 85% of the cross

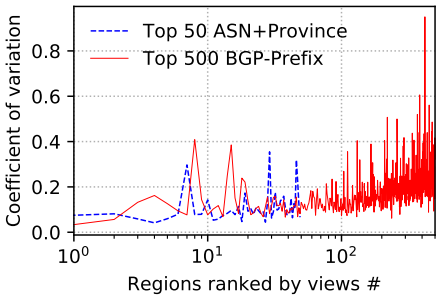


Fig. 24. How variable are everyday' s M values for server locations (top 50 ASN+Provinces or top 500 BGP-Prefixes).

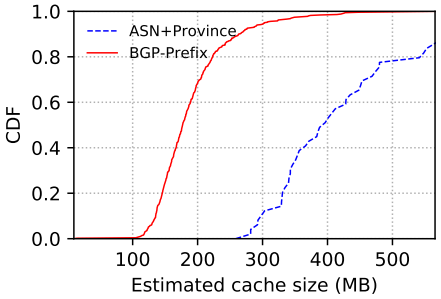


Fig. 25. Estimated cache server storage space requirement for server locations.

ASN+Province views can be improved via pre-fetching. The median R over all network regions is as high as 29.5%.

Storage Overhead: Finally, we evaluate the storage cost introduced by pre-fetching content on the regional cache servers. Assume that at some time point, t_i, an edge server s receives several viewing requests for n_{t_i} different broadcasts, and $M = max\{n_{t_i}\}$ for all observed time points in a day. Since only the latest GoP of each broadcast is cached,[6] the maximum storage space required by s can be approximated by $M \times c$, where c is the size of a GoP[7] and is set to 0.5MB [44].

To examine the upper bound of M, we first analyze the variation of M for each server location (top 50 ASN+Provinces or top 500 BGP-Prefixes). To this end, we compute the coefficient of variation (cv for short) for each day's M value of each server location; the results are shown in Figure 24. We observe a small cv for most server locations: those top-rank server locations barely vary, with extremely small cv (< 0.1). The median cv value is only 0.09 for ASN+Province, and 0.17 for BGP-Prefix.

Given the low variation of each day's M values for individual servers, we then proceed to estimate M's upper bound of M_{bound} for each server as follows. Assume that each day's M values of a server follow a Gaussian distribution, with the average $\bar{M}$ and standard deviation δ. Then $\bar{M} + 2.58\delta$ can meet the storage requirement in 99% of the cases [11]. Considering the influence of the observed M values, $M_{bound} = max\{\bar{M} + 2.58\delta, max(M)\}$. We can

6. There is no benefit in storing older GoPs for live streams.
7. We assume c is a constant value.

TABLE 5
Benefits of different configurations of the peer-assisted delivery.

	ASN+Province	BGP-Prefix
SLR (server load)	80.0%	51.5%
CTR (cross-region)	79.8%	51.5%

then estimate the storage required of the cache server as $M_{bound} \times c$. We apply the above estimation method to each cache server and show the results in Figure 25. The estimated maximum storage space for nearly all servers is under 1GB. We argue this is highly feasible for most streaming systems, and highlights the benefit of restricting caching to only recent GoPs.

5.4 Evaluation of Peer Assistance

To explore the benefit of EDGEOPT's peer-assisted delivery scheme, we develop an event-driven simulator for peer-assisted live streaming (similar to BitTorrent), where we obtain the upload capacity of a peer using the observed mean video bitrate of broadcasters in the same ASN+Province as the peer. Based on the empirical statistics from our dataset, we set the time taken for viewers to transfer one GoP to another peer to 1 second. Note that in our simulation, we try to redirect each individual view to a proper peer, regardless of the utilization of edge servers, in order to estimate the maximum potential.

With the above settings, we replay the service logs of Medium and Heavy broadcasters through our simulator (we exclude low popularity broadcasters, since they have too few viewers). We focus on two metrics: (*i*) Server Load Reduction (SLR), which represents the reduction in download traffic handled by edge servers; and (*ii*) Cross-region Transmission Reduction (CTR), which captures how much download traffic originally coming from servers in different regions to the viewer is now conveyed by local peers in the same region.

We present the results of these two metrics in Table 5. We see a significant reduction in server load and cross-region transfers. Even when limiting clients to transferring to only one peer in the same BGP-Prefix, the server load is halved. This is because of the localization of viewers within a few network regions (§3.3). In addition, allowing clients to connect to peers in the same ASN+Province results in substantially better performance than restricting them to the same BGP-Prefix, as it allows clients to select from a larger pool of peers to download from. One may also notice that the values of SLR and CTR are close. This is because, originally, most of the data is transferred cross-regionly. Hence, reductions in server load naturally result in reductions in cross-region traffic.

5.5 Discussion

The decision tree model (for predicting the broadcasts' popularity) and the DNN model (for forecasting the viewer arrival time) capture the usage patterns of MLS, which are likely to be stable in the medium-term, *e.g.*, one week. This has also been partially proved by our trace-driven experiments, where the training sets are from data of the first five/six days (out of ten days), and we see that both models perform well on the test set (*i.e.*, the data of the following five/four days). Therefore, we posit that a weekly update is appropriate.

6 RELATED WORK

MLS User Behavior Analysis: The behavioral patterns of MLS and general live streaming systems have been examined in several works. Raman *et al.* [35] explored the video characteristics, and the social engagement of Facebook Live, and highlighted many unwatched broadcasts. Li *et al.* [24] characterized the user behavior in the PPTV mobile live streaming system, and they examined the structure of the abandoned session problem. Ma *et al.* [28] investigated Inke Live, and identified differences between MLS and conventional live services. Periscope and Meerkat were examined in [37], [39], [41] from the perspectives of latency and usage patterns. There have been several studies of the Twitch live streaming system [15], [16], [47].

Our measurements differ from the above studies in two major ways: (*i*) We explore unwatched broadcasts in depth for the first time, and find that the greatest source of waste is partially unwatched streams (as opposed to the unwatched broadcasts); and (*ii*) We identify localized viewership at the BGP-Prefix level, which underpins our subsequent edge server pre-fetching.

MLS System Optimizations: While we focus on live broadcast uploads, resource wastage in the on-demand video has been examined in [48]. The authors proposed a post-streaming wastage analysis algorithm to achieve the best tradeoff between QoE and resource-saving. By using selective quality-enhancing retransmission, Ray *et al.* [36] presented a live broadcast upload framework that improves the overall QoE for time-shifted viewers. The pre-fetching technique is investigated in [34], where an app prediction system is designed, predicting which app will be used next and ensuring the freshness of the content. In addition, Li *et al.* [25] enhance video quality by localizing video delivery through caching. Yan *et al.* [43] explore the potential of caching contents in a CDN by learning from an optimal caching allocation. There have also been several proposals for peer-assisted delivery [50] [26] of (mobile) VoD.

Various other video optimizations have been proposed. Ghabashneh *et al.* [18] conducted a measurement study on how CDN caches interplay with the viewing experience. Based on their observations, they proposed an ABR algorithm incorporated with CDN awareness. Wang *et al.* [42] proposed a reinforcement learning based scheme deployed at the edge CDN server, to dynamically select a suitable initial video segment for new live viewers, to optimize viewer QoE. Zhang *et al.* [49] presented a super resolution based adaptive video streaming framework, which allows clients to download low bitrate video segments, reconstruct and enhance them to high-quality video segments. Siekkinen *et al.* [38] provided close to optimal algorithms for scheduling video chunk uploading for multiple clients with different viewing delays. There have also been several studies on improving Adaptive Bit Rate (ABR) in mobile networks [30], [32], [45], [46]. These studies leverage cross-layer designs that consider both video content features and

underlying mobile network or channel information for QoE optimization.

In contrast to these prior studies, we focus on optimizing the delivery on the edge for both the first-mile and last-mile transmission using a data-driven approach. Through the design of EDGEOPT, we primarily aim to improve resource usage while improving video QoE.

7 CONCLUSIONS AND FUTURE WORK

This paper has presented a detailed analysis of MLS user behavior. The main findings include redundant uploads, highly localized viewer locality, and the predominance of predictable loyal viewers. Based on these insights, we have proposed EDGEOPT to optimize the MLS system on the edge from two perspectives. First, to improve upload resource utilization in the first-mile, a decision tree based system has been proposed to reduce the encoding rate of wasteful uploads. We then reallocate bandwidth resources to a high-light retransmission scheme to enhance the replay quality. Second, to streamline the delivery in the last mile, EDGEOPT adopts a learning-based pre-fetching scheme for reducing the startup delay, and a peer-assisted delivery for alleviating edge server loads. Our trace-driven experiments have shown EDGEOPT's efficacy in boosting resource utilization as well as enhancing QoE.

There are a number of future work directions. Although EDGEOPT mainly focuses on improving resource utilization and startup latency, we note that the end-to-end live streaming latency is also a key metric for QoE optimization. This is especially the case for interactive live streaming, where the broadcaster and viewers should be able to communicate in real time. We therefore wish to expand EDGEOPT with solutions to improve the live latency. In addition, the peer-assisted delivery now assumes a viewer can either get the data from a peer or from the edge server (but not both). To overcome this, we plan to explore techniques such as network coding to enable multiple source delivery.

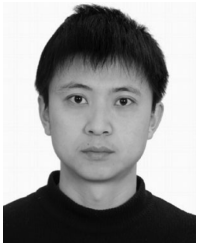
ACKNOWLEDGMENT

This work was partially supported by Natural Science Foundation of China (U20A20180, 62072437), Beijing Natural Science Foundation (JQ20024), and CAS-Austria Joint Project (171111KYSB20200001).

REFERENCES

[1] Wowza's dash bitrate recommendation. https://www.wowza.com/docs/howto-encode-source-video-for-wowza-streaming-cloud.
[2] Cisco: 2020 global networking trends report. https://www.cisco.com/c/m/en_us/solutions/enterprise-networks/networking-report.html#, 2020.
[3] Robustscaler - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html, 2020.
[4] Crypto-pan. https://www.cc.gatech.edu/computing/Networking/projects/cryptopan/, 2022.
[5] Facebook live. https://live.fb.com/, 2022.
[6] K-means - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html, 2022.
[7] Pca - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, 2022.
[8] Periscope. https://www.periscope.tv/, 2022.
[9] Standardscaler - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, 2022.
[10] Twitch. http://www.twitch.tv/, 2022.
[11] D. G. Altman and J. M. Bland. Standard deviations and standard errors. *Bmj*, 331(7521):903, 2005.
[12] Z. Avramova, D. De Vleeschauwer, K. Spaey, S. Wittevrongel, H. Bruneel, and C. Blondia. Comparison of simulcast and scalable video coding in terms of the required capacity in an iptv network. In *Packet Video 2007*, pages 113–122, 2007.
[13] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3):301–315, 1998.
[14] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo. Analysis and design of the google congestion control for web real-time communication (webrtc). In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–12, 2016.
[15] J. Deng, F. Cuadrado, G. Tyson, and S. Uhlig. Behind the game: Exploring the twitch streaming platform. In *2015 International Workshop on Network and Systems Support for Games (NetGames)*, 2015.
[16] J. Deng, G. Tyson, F. Cuadrado, and S. Uhlig. Internet scale user-generated live video streaming: The twitch case. In *International Conference on Passive and Active Network Measurement*, pages 60–71. Springer, 2017.
[17] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.
[18] E. Ghabashneh and S. Rao. Exploring the interplay between cdn caching and video streaming performance. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 516–525, 2020.
[19] A. Halfaker, O. Keyes, D. Kluver, J. Thebault-Spieker, T. Nguyen, K. Shores, A. Uduwage, and M. Warncke-Wang. User session identification based on strong regularities in inter-activity time. In *Proceedings of the 24th International Conference on World Wide Web*, pages 410–418, 2015.
[20] S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking*, 21(6):2001–2014, 2013.
[21] L. Leydesdorff. On the normalization and visualization of author co-citation data: Salton's cosine versus the jaccard index. *Journal of the American Society for Information Science and Technology*, 59(1):77–85, 2008.
[22] J. Li, Z. Li, R. Lu, K. Xiao, S. Li, J. Chen, J. Yang, C. Zong, A. Chen, Q. Wu, C. Sun, T. Gareth, and H. H. Liu. Livenet: A low-latency video transport network for large-scale live streaming. In *Proceedings of the 2016 ACM SIGCOMM Conference*, 2022.
[23] J. Li, Z. Li, Q. Wu, and G. Tyson. On uploading behavior and optimizations of a mobile live streaming service. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022.
[24] Z. Li, M. A. Kaafar, K. Salamatian, and G. Xie. Characterizing and modeling user behavior in a large-scale mobile live streaming system. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2675–2686, 2016.
[25] Z. Li, Q. Wu, K. Salamatian, and G. Xie. Video delivery performance of a large-scale vod system and the implications on content delivery. *IEEE Transactions on Multimedia*, 17(6):880–892, 2015.
[26] J. Lin, Z. Li, G. Xie, Y. Sun, K. Salamatian, and W. Wang. Mobile video popularity distributions and the potential of peer-assisted video delivery. *IEEE Communications Magazine*, 51(11):120–126, 2013.
[27] W.-Y. Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.
[28] M. Ma, L. Zhang, J. Liu, Z. Wang, W. Li, G. Hou, and L. Sun. Characterizing user behaviors in mobile personal livecast. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 43–48, 2017.
[29] T. Martin, J. M. Hofman, A. Sharma, A. Anderson, and D. J. Watts. Exploring limits to prediction in complex social systems. pages 683–694, 2016.
[30] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski. Edge computing assisted adaptive mobile video streaming. *IEEE Transactions on Mobile Computing*, 18(4):787–800, 2019.

[31] N. Moniz and L. Torgo. A review on web content popularity prediction: Issues and open challenges. *Online Social Networks and Media*, 12:1–20, 2019.
[32] P. K. Mu, J. Zheng, T. H. Luan, L. Zhu, M. Dong, and Z. Su. Amis: Edge computing based adaptive mobile video streaming. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10, 2021.
[33] H. Pang, Z. Wang, C. Yan, Q. Ding, and L. Sun. First mile in crowdsourced live streaming: A content harvest network approach. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pages 101–109, 2017.
[34] A. Parate, M. Böhmer, D. Chu, D. Ganesan, and B. M. Marlin. Practical prediction and prefetch for faster access to applications on mobile phones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, page 275–284, New York, NY, USA, 2013. Association for Computing Machinery.
[35] A. Raman, G. Tyson, and N. Sastry. Facebook (a) live?: Are live social broadcasts really broad casts? In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1491–1500. International World Wide Web Conferences Steering Committee, 2018.
[36] D. Ray, J. Kosaian, K. Rashmi, and S. Seshan. Vantage: optimizing video upload for time-shifted viewing of social live streams. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 380–393. 2019.
[37] M. Siekkinen, E. Masala, and T. Kämäräinen. A first look at quality of mobile live streaming experience: the case of periscope. In *Proceedings of the 2016 Internet Measurement Conference*, pages 477–483, 2016.
[38] M. Siekkinen, E. Masala, and J. K. Nurminen. Optimized upload strategies for live scalable video transmission from mobile devices. *IEEE Transactions on Mobile Computing*, 16(4):1059–1072, 2017.
[39] J. C. Tang, G. Venolia, and K. M. Inkpen. Meerkat and periscope: I stream, you stream, apps stream for live streams. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4770–4780, 2016.
[40] Y. Tian, R. Dey, Y. Liu, and K. W. Ross. Topology mapping and geolocating for china's internet. *IEEE Transactions on Parallel and Distributed Systems*, 24(9):1908–1917, 2012.
[41] B. Wang, X. Zhang, G. Wang, H. Zheng, and B. Y. Zhao. Anatomy of a personalized livestreaming system. In *Proceedings of the 2016 Internet Measurement Conference*, pages 485–498, 2016.
[42] 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 - IEEE Conference on Computer Communications*, pages 706–715, 2020.
[43] G. Yan, J. Li, and D. Towsley. Learning from optimal caching for content delivery. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '21, page 344–358, New York, NY, USA, 2021. Association for Computing Machinery.
[44] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han. Nemo: Enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, New York, NY, USA, 2020. Association for Computing Machinery.
[45] H. Yuan, H. Fu, J. Liu, J. Hou, and S. Kwong. Non-cooperative game theory based rate adaptation for dynamic video streaming over http. *IEEE Transactions on Mobile Computing*, 17(10):2334–2348, 2018.
[46] A. H. Zahran, D. Raca, and C. J. Sreenan. Arbiter+: Adaptive rate-based intelligent http streaming algorithm for mobile networks. *IEEE Transactions on Mobile Computing*, 17(12):2716–2728, 2018.
[47] C. Zhang and J. Liu. On crowdsourced interactive live streaming: a twitch. tv-based measurement study. In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 55–60, 2015.
[48] G. Zhang, K. Liu, H. Hu, V. Aggarwal, and J. Lee. Post-streaming wastage analysis a data wastage aware framework in mobile video streaming. *IEEE Transactions on Mobile Computing*, pages 1–1, 2021.
[49] Y. Zhang, Y. Zhang, Y. Wu, Y. Tao, K. Bian, P. Zhou, L. Song, and H. Tuo. Improving quality of experience by adaptive video streaming with super-resolution. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 1957–1966, 2020.
[50] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec. Peer-assisted content distribution in akamai netsession. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, 2013.
[51] Z. Zheng, Y. Ma, Y. Liu, F. Yang, Z. Li, Y. Zhang, J. Zhang, W. Shi, W. Chen, D. Li, Q. An, H. Hong, H. H. Liu, and M. Zhang. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 418–432, 2021.



Zhenyu Li received the BS degree from Nankai University in 2003 and the PhD degree in Graduate School of Chinese Academy of Sciences (CAS) in 2009. He is a professor at the Institute of Computing Technology, CAS. His research interests include Internet measurement and Networked Systems.

Jinyang Li is currently a Ph.D candidate at the Institute of Computing Technology (ICT), Chinese Academy Sciences (CAS). He received his B.S. degree in Computer Science from Sichuan University, Chengdu, China, in 2017. His research interests include low-latency transmission, and Internet measurements.

Qinghua Wu received the Ph.D. degree from ICT/CAS in 2015. He is currently an Associate Researcher at ICT/CAS. His research interests lie in network transport protocol and Internet measurements.

Gareth Tyson received the PhD degree from Lancaster University in 2010. He is an Assistant Professor at Hong Kong University of Science and Technology (GZ). His research interests include Internet measurements and content distribution.

Gaogang Xie received the PhD degree in computer science from Hunan University, in 2002. He is a professor in the Computer Network Information Center, Chinese Academy of Sciences. His research interests include Internet architecture, SDN/NFV, and Internet measurement.