On Uploading Behavior and Optimizations of a Mobile Live Streaming Service

Jinyang Li*†, Zhenyu Li*†§, Qinghua Wu*†§, Gareth Tyson‡¶
*Institute of Computing Technology, Chinese Academy of Sciences, †University of Chinese Academy of Sciences,
‡Hong Kong University of Science & Technology, §Purple Mountain Laboratories, China,
¶Queen Mary University of London
{lijinyang,zyli,wuqinghua}@ict.ac.cn, gareth.tyson@qmul.ac.uk

***Abstract*—Mobile Live Streaming (MLS) services are now 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 (*e.g.,* LTE). Although prior work has focused on *viewer-side* behavior, it is equally important to study and improve *broadcaster* operations. Using detailed logs obtained from a major MLS provider, we first conduct an in-depth measurement study of uploading behavior. Our 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 to a small set of broadcaster-specific network regions. Inspired by our findings, we propose two system innovations to streamline MLS systems: adaptive uploading and edge server pre-fetching. These optimizations leverage machine learning for reduced waste and improved QoE. Trace-driven experiments show that the adaptive uploading reduces the resources wastage by 63%, and the pre-fetching boosts the startup by 29.5%.**

I. Introduction

The majority of Internet traffic is now video [1], and with the development of mobile devices, Mobile Live Streaming (*MLS*) services like Facebook Live [3], and Periscope [6] 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), but also helps build an era of "live broadcast for the people".

Due to its novelty, MLS has received increasing attention from the research community. For example, there has been work on adaptive bitrate streaming for MLS [35], [27], [15], and active measurements [28], [34]. However, most efforts have focused on the *last-mile* (from server to viewer) instead of the *first-mile* (from broadcaster to server). As MLS broadcasters are often amateurs, and in contrast to other streaming systems, this means that both the last and first mile are often unreliable (*e.g.,* 3G/4G), thereby creating potential consequences for overall Quality of Experience (QoE) [25]. This challenge is exacerbated by the unpredictable nature of user-generated uploads, making capacity management more difficult. We argue that we need a more refined understanding of MLS, as well as new techniques for managing its unique properties.

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. To the best of our knowledge, this work is among the first to examine and optimize the MLS broadcaster side. Our measurements reveal three key observations (§III):

- ***Wasteful Uploads:*** We identify significant volumes of wasteful live video uploads (33.3% of total upstream traffic). Waiting for the first viewer's arrival and viewer clear-outs during streaming are the two dominant contributors; these two wastes constitute 30% of total upstream traffic. Besides, the streams are unlikely to be 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. Besides, cross-region delivery of content to viewers doubles the startup delay, compared with same-region delivery.
- ***Loyal Viewers:*** We find that "loyal" viewers, who regularly watch the same broadcasters, are small in number but generate 59% of the total video data downloaded. Perhaps more importantly, these loyal viewers not only arrive earlier but also leave later.

Based on the above insights, we optimize the MLS system from two perspectives (§IV): (*i*) *Reducing Waste*: To alleviate upload waste caused by unwatched segments, we propose a decision tree based adaptive uploading system, which relies on 70 features for predicting the attractiveness level of the content that follows individual clear-outs. As a result, the resource wastage is reduced by 63%. (*ii*) *Boosting Viewing Experience*: We propose an edge server pre-fetching strategy, composed of a pre-fetching location selection scheme based on our observations of viewing locality and loyal viewers, and a pre-fetching timing prediction scheme using a deep neural network. Through this, we reduce startup delay by 29.5%.

II. Background & Dataset

A. Overview of MLS

Our data comes from a large-scale Mobile Live Streaming (MLS) broadcasting platform that serves millions of users per day. Anyone on such a platform can be a broadcaster, streaming their camera feed. When a user starts broadcasting, they upload video segments to one of the ingest servers, which in turn re-encodes them at various bit rates and then publishes chunks to a Content Delivery Network (CDN). The CDN is then responsible for disseminating (appropriately encoded) segments to viewers who request content via HTTP.

B. Dataset Description

We rely on over 1.8TB of logs, covering 10 days of anonymous access shared by the examined 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 of video. 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, BGP-Prefix, ASN, and province-level geographical location.[1]
2) *Session-specific:* Data volume, duration of the session, direction (upload or download).
3) *Viewer-side QoE:* Various QoE metrics, including startup delay, number of buffering events, buffering duration, and the number of retries.
4) *Operating environments:* Network connection type (*e.g.,* 4G or WiFi), platform type (*e.g.,* Android or iOS).

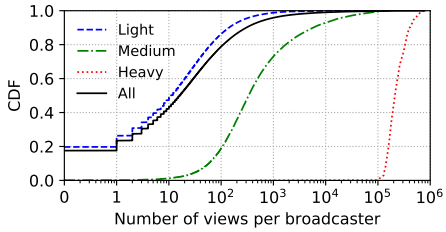
Although the examined MLS also supports desktop access, 99+% of both the broadcasting and viewing traffic contained in our dataset is from mobile devices.

Ethical Considerations: We took a number of steps to ensure 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.

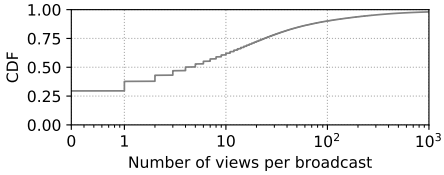
III. Uploading Behavior Analysis

A. 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. Despite this, we observe that the App on the broadcaster side keeps uploading content regardless, thereby *wasting* resources without anybody viewing the content. We find that in total 33.3% of upstream traffic is wasted (see §III-B for breakdown analysis). This suggests the potential for streamlining the delivery mechanism, by lowering down video quality for unwatched segments. We present such an optimization in §IV-A.



(a) Sum number of views per broadcaster



(b) Number of views per broadcast

Fig. 1: Number of views received by per broadcaster and broadcast. We later define *Light, Medium, Heavy* broadcaster groups (Table I).

To get a handle on why this might happen, we look into the viewing patterns of consumers. Unsurprisingly, the platform is dominated by short-form content. Viewing times tend to be short (median ≈10s), and only a tiny fraction (median ≈0.1%) of the overall broadcast will be consumed in a viewing session. Despite this, most viewers watch a large number of distinct videos. Within the examined 10 days, the median view count per viewer is 18, and 22% of viewers watch at least 100 live broadcasts. Thus, we conjecture that many viewers skip between videos, looking for content of interest. We find that 76% of intervals between two consecutive views are under 1 second, indicating this is indeed the case.

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 [8] and Principal Component Analysis (PCA) [5] to preprocess the data, on which we apply K-Means [4], experimenting with K from 3 to 10. We select K=3 due to a relatively small Davies-Bouldin Index [10].

Table I presents the results. The broadcasters can be divided into 3 main categories: *Light* (L), *Medium* (M), and *Heavy* (H).

[1]The anonymization is done by `Crypto-PAn` [2]

TABLE I: 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

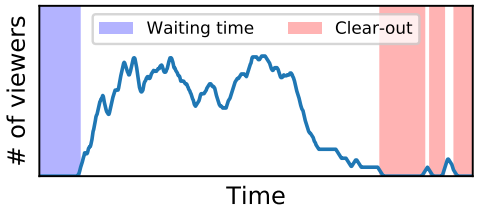


Fig. 2: An illustration of *waiting time* and *clear-out*.

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 number of medium active broadcasters is relatively small, but the degree of activity and popularity is far more than the Light cluster. The most active/popular broadcasters (Heavy) are distinct from the other two types though: They are the *"stars"* of the platform. Although the number of these broadcasters is small, they contribute a disproportionately large amount of upstream and downstream traffic.

B. 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 kinds of unwatched broadcast *segments* (*i.e.,* partially unwatched), which we examine for the *first* time: (*i*) 7% of upstream traffic is wasted while waiting for the first viewer to arrive (termed *waiting time*); (*ii*) 23% of upstream traffic is wasted because all viewers exit before the broadcast ends (termed a *clear-out*). For illustration, Figure 2 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.

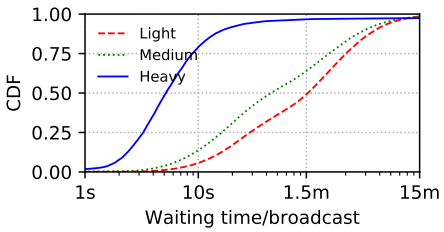


Fig. 3: Waiting time distribution for each type of broadcaster.

Waiting for the First Viewer: Overall, 7% of upstream traffic is wasted because contents are uploaded before the first viewer's arrival. Figure 3 presents the waiting time distribution for broadcaster groups. We find that, in most Heavy group 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.

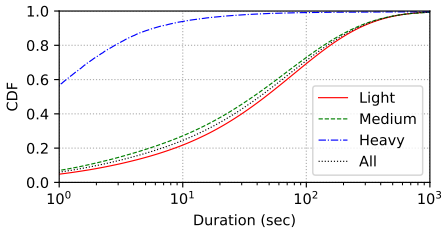
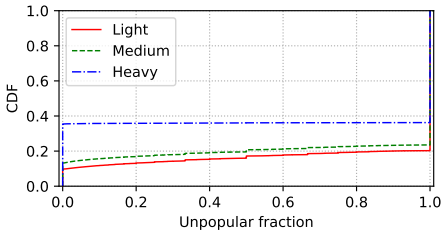
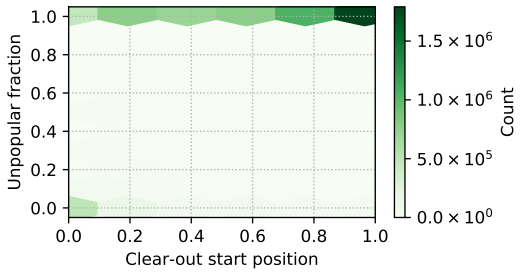


Fig. 4: Clear-out duration distribution for each type of broadcaster.

Clear-out Periods: 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 4 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 in Heavy broadcasters never exceed 1 second, indicating that the clear-out length is related to popularity. In addition, clear-outs are likely to occur more than once per broadcast (4 times median).



(a) Unpopular fraction distribution



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

Fig. 5: Unpopular fraction is related with broadcaster group and where the clear-out happens.

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

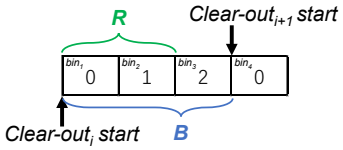


Fig. 6: An illustration of unpopular fraction calculation.

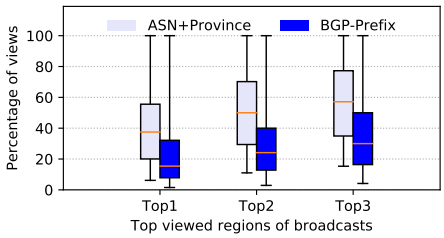
likely to be closely followed by another one. This observation indicates that, once a broadcast has started to lose attraction (*i.e.,* experiencing several clear-outs), it is unlikely to regain it. To confirm this, we split the broadcast between the starting points of clear-out$_i$ and clear-out$_{i+1}$ 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 6. We then record the run length (R) of unpopular bins (whose $v \leq 1$). Finally, we define *unpopular fraction* f as $f = \frac{R}{B}$. For example, in Figure 6, $B = 3$, $R = 2$, and thus $f = 2/3$. The larger the value of f, the less popular the content between the two clear-outs is. We present the distribution of f in Figure 5(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 5(b), where 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 L/M/H broadcast, respectively). That said, the clear-outs occurring in the late stage of broadcasts indicate the content is losing attraction, and thus can be suppressed to save traffic. We further confirm this implication in §IV-A, with a decision tree model.

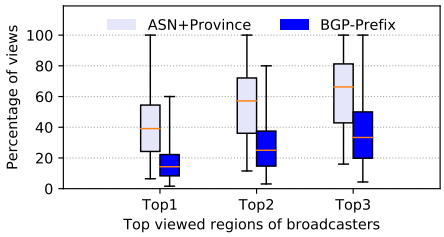
C. Mobile Characteristics

Usage Patterns Among Connection Types: We next examine the usage patterns of broadcasting and viewing among different network connection types (*e.g.,* WiFi/cellular). We observe a WiFi-dominated system, where WiFi carried 92.9% of the uploading traffic. Broadcasters using WiFi upload for longer, and produce more streams (compared with their cellular counterparts).

For viewing, interestingly, we observe that viewers regularly switch their connection types in the middle of the viewing



(a) Broadcast



(b) Broadcaster

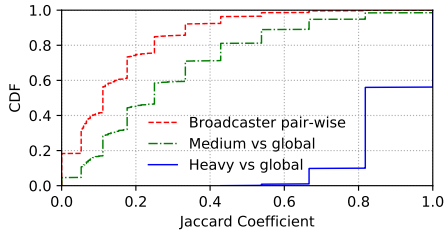
Fig. 7: Percentage of views contributed by the top three regions.

(*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 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%). Curiously, although these connection-switching views experience the worst startup delay (27ms *vs.* an overall average of 16ms), their viewing sessions 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.

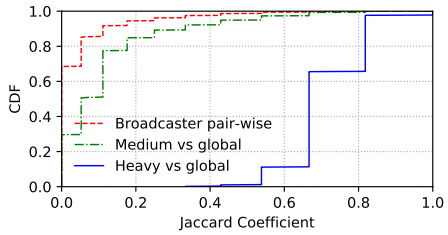
View Locality: Next, we study the locality patterns of viewers. This is important if edge caches were to be deployed. Since we are interested in users' network footprint, we use the BGP-Prefix and ASN+Province[2] of the user's IP address to represent her location. Note that we use the ASN+Province combination, as a given ASN may have a presence in multiple provinces.

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 7 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

[2]China's Internet follows a hierarchical structure. For each ISP, provinces manage their own regional networks [33].

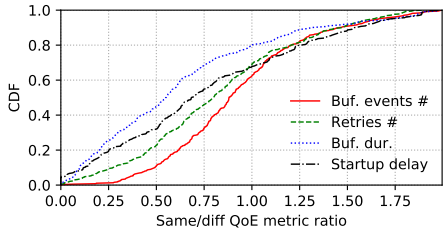


(a) ASN+Province

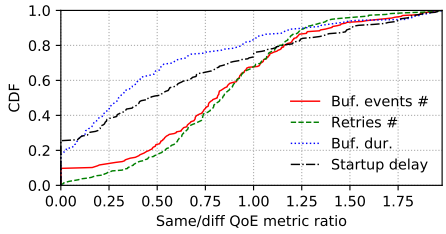


(b) BGP-Prefix

Fig. 8: Overall small Jaccard coefficient indicate broadcaster-specific locality.



(a) ASN+Province



(b) BGP-Prefix

Fig. 9: Distribution of $\frac{same\ region}{cross\ region}$ QoE metrics in ASN+Province networks (a) and BGP-prefix networks (b).

viewing patterns. 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.

We next 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* [17] for every pair of broadcasters (r_i, r_j). A higher coefficient implies more overlap between the top regions of the two broadcasters. Figure 8 plots the distribution of the coefficient.[3] 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 8). 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 (§IV-B).

We also observe that despite the viewing locality, the majority of streams (from servers to viewers) cross network boundaries: 88.6% of the viewing data is transmitted across ASN+Province networks, whereas 99.8% of broadcasts stream to a different BGP-Prefix.

Impact on QoE: The above indicates significant scope for localizing traffic via techniques such as pre-fetching. To understand the potential QoE benefits, we calculate four relevant viewer metrics: (*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 CDN servers 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 9 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 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 the same ASN+Province case, which is expected as BGP-Prefix is a smaller network region. This confirms that bringing serving content closer to consumers (*e.g.,* pre-fetching) could significantly improve QoE.

D. 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 without reliance on explicit information (*e.g.,* follower list [21]).

Identifying Loyal Viewers: To determine whether viewer v who has watched broadcaster b's streams is a loyal viewer of

[3]We excluded Light broadcasters to prevent from randomness.

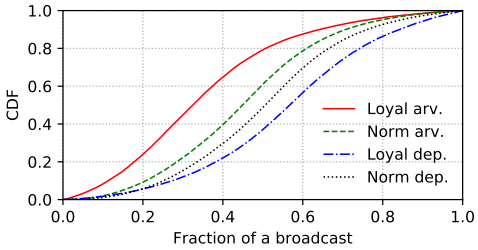


Fig. 10: The position of a broadcast where loyal/normal viewers arrive/depart.

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.

TABLE II: 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%

The clustering results are shown in Table II. The viewer/broadcaster pairs can be 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 most of the downloading: 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 loyal viewers would be disproportionately beneficial.

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 10. 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 (§III-A), loyal viewers consume persistently. As such, we posit that loyal viewers are potentially good candidates for stable peers in peer-assisted delivery [19].

E. Takehome Messages

Redundant Uploads: A significant fraction of video content is uploaded 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.,* waiting time and clear-out). Thus, suppressing redundant uploads of unwatched content could mitigate the traffic load (§IV-A). In particular, the clear-outs that happen in the late stage of individual streams are a good indicator of starting suppression.

Predictable Locality Traits: Broadcasts show strong localized viewing attraction. 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 the network boundaries, doubles the startup delay and buffering duration in 50% of cases. Techniques such as end server pre-fetching to localize the delivery would therefore have a great potential to improve viewers' QoE (§IV-B).

Loyal Viewers' Contributions: Some popular broadcasters are viewed regularly by loyal viewers. Although the number of loyal viewers is small, they consume the majority of download volume (59%). Further, loyal viewers come to the broadcast channel earlier and also leave later than others. This makes loyal viewers easy to predict for pre-fetching (§IV-B).

IV. System Optimizations

A. Adaptive Uploading

Design: We have found that uploads with no viewers during the clear-out periods cause 23% of broadcast traffic to be wasted. Meanwhile, we find that most clear-outs (*i.e.,* those happening in the late stage of streams) are a sign that broadcasts have lost attraction to viewers (§III-B). To alleviate this resource wastage, we propose an upload bit rate control scheme based on the *attractiveness level* of broadcasts. Specifically, every time a clear-out occurs, we use a trained model to predict the "attractiveness" of the broadcast in the future. Thus, we treat this as a classification task. If the broadcast is predicted to be "unattractive", then it will upload at a low bit rate (*e.g.,* 144p) until a viewer arrives, at which time the bit rate will switch to normal. We consider a broadcast as unattractive if its *unpopular fraction*, f (defined in §III-B) equals 1. Put simply, an unattractive live broadcast will *not* have more than one viewer simultaneously until the next clear-out occurs.

We choose a decision tree [20] as our classification model due to its efficiency and interpretability. Moreover, the white-box nature of decision trees will help system operators to

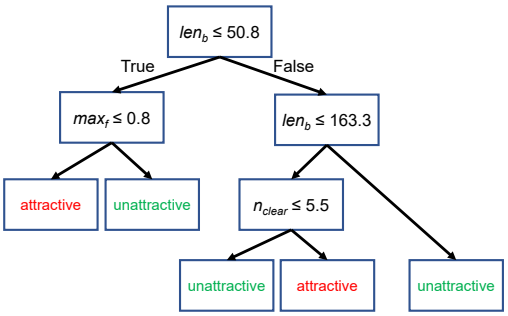


Fig. 11: The decision tree trained for clear-out classification. len_b is the length of the broadcast so far; n_{clear} means the number of clear-outs that have occurred in the broadcast; max_f is the maximum f of the broadcast so far.

adjust the system [23]. For classification, 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 occurred in the broadcast (n_{clear}) (*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.

Results: We use 60% of the clear-out statistics for training and the rest for testing. After training, the depth of the unpruned decision is 245. We further exploit Cost Complexity Pruning (CCP) [20] to prune the decision tree.

The resulting tree structure is presented in Figure 11. It achieves 85% precision, 96% recall and the F1 score is 0.9. 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 (§III-B).

After we replay the viewing records from our dataset on the simulation system using the above tree model, we save as high as 63% of the data volume during the clear-out period. This accounts for 14.5% of the total uploading traffic.

Potential Impact on Viewing Duration: A risk of the adaptive uploading scheme is the arrival of viewers will result in them initially experiencing poorer QoE due to the low bit rate. To understand the impact of this on viewing duration (as a proxy of viewer engagement), we adopt a *Quasi-Experiment Design* (QED) [29]. This is a popular technique in social science and has been previously used to study QoE [9].

In QED, we control several variables and study the cause-effect relationships between the broadcast's *initial* bit rate and the viewing duration within our dataset. To do this, we compare two randomly selected broadcasts (u, v) with the same values of the controlled variables. Specifically, we control for the broadcaster's device type (iOS/Android) and the streamer's network type (4G/WiFi). We then assign broadcasts into bins, such that all broadcasts in a bin have the same values for the control factors.

To provide causal evidence for the initial bit rate's potential impact on viewing duration, we define *outcome* [16] for each (u, v) pair as follows. $outcome(u, v) = +1$ if the broadcast with the higher initial bit rate gains the longer view duration, compared with the other one; $outcome(u, v) = -1$, otherwise. Then, we compute $Net\ Outcome$ for each bin as follows:

$$Net\ Outcome = \frac{\sum_{(u,v)\in bin} outcome(u, v)}{|bin|} \quad (1)$$

where bin is the set of pairs in the examined bin. Note, a positive value for $Net\ Outcome$ provides positive (supporting) evidence for the causality, while a negative value provides negative evidence; a value close to 0 implies less or no causality. Overall, the $Net\ Outcome$ is 0.06, which shows that a higher initial bit rate results in a longer viewing session, but the impact is very limited.

To measure the significance of the above QED result, we further perform hypothesis testing, in which the null hypothesis, H_0 states that the high initial bit rate has *no* impact on view duration. Then, we use the sign test results for pairs ($outcome(u, v)$) to derive a bound on the p-value by computing the two-sided tail of the binomial distribution with n trials and probability 1/2, where n is the number of pairs. Our result shows a low p-value: 1.17×10^{-14}, which is much smaller than the required significance level of 0.001, indicating a rejection of H_0. Thus, our QED analysis is statistically significant. This indicates that introducing short periods of low-resolution video will not have a major impact on viewer retention. Note that only the first few seconds of the first viewer will be impacted by the low initial bit rate, which will be soon repaired by techniques like ABR. Indeed, a conservative initial bit rate is common in modern video systems (*e.g.,* Netflix) for preventing high startup delay [31].

B. Edge Server Pre-Fetching

High video startup delays negatively impact viewer QoE [16], [13]. In §III-C we observed geographically localized viewing patterns, indicating that there is scope to reduce start-up delays by placing content in consumers' locales. Hence, we next propose a pre-fetching scheme to preemptively retrieve content predicted to be viewed in a given locale.

Overview: We assume that viewers in each region (ASN+Province or BGP-Prefix) share a local cache server. For the purposes of our experiments, We experiment with two setups: placing edge servers in (*i*) the top 50 (1%) ASN+Provinces, or (*ii*) top 500 (5%) BGP-Prefixes. We term locales equipped with a cache as *server locations*. 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), according to the pre-fetching strategy that is described below. Here, a GoP is set to 120 frames [37].

There are three key challenges in the above 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: We strive to pre-fetch the most popular content. Here, we rely on the historical popularity of broadcasters, as this is a good predictor of future popularity. Hence, we pre-fetch *all* broadcasts of Medium/Heavy broadcasters as

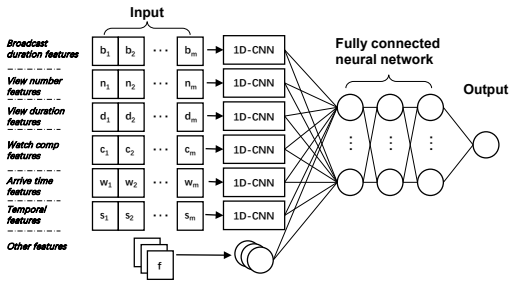


Fig. 12: The DNN model for arrival time prediction.

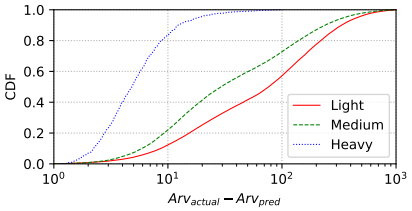


Fig. 13: CDF of $error = Arv_{actual} - Arv_{pred}$ for each broadcaster type. The x-axis is in second.

they consistently accumulate large audiences (see Table I). Specifically, Medium/Heavy broadcasters, who only make up 10% of uploader popularity, disproportionately produce 90% of the total viewership.

Where To Pre-Fetch: We propose an edge server selection strategy as follows. For each broadcaster, we select the top k regions from edge server locations where viewers generate the most historical views on their broadcasts. The broadcaster's stream will then be forwarded directly to these k server locations. Clients wishing to consume a stream then send all 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. For comparison, we use a baseline strategy where contents will be pre-fetched to the global top k regions.

When To Pre-Fetch: We next investigate how to determine the proper time to pre-fetch. This is important, since pre-fetching too early (*i.e.,* long before viewers' arrival) will bring no positive influences and only waste bandwidth resources, since no one is watching the uploaded content. Our objective is to ensure the predicted arrival time, Arv_{pred}, is close to the actual arrival time, Arv_{actual}, reserving sufficient time for the content to be relayed from the source server to the edge before viewers' arrival. Formally, Arv_{pred} and Arv_{actual} should satisfy the following restrictions:

$$Arv_{actual} - Arv_{pred} - t_{relay} \geq 0 \tag{2}$$

$$Arv_{actual} - Arv_{pred} - t_{relay} \leq \epsilon \tag{3}$$

where ϵ is a very small 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, we propose a deep neural network based regression model, whose architecture is presented in Figure 12. For each broadcast we try to predict, we gather 70 features in 5 categories from its broadcaster's past 5-day of activity. Due to the page limit, we only brief each category with its representative features: (*i*) *Broadcaster-specific*: Number of broadcasts, broadcaster type (*i.e.,* Light, Medium or Heavy); the number of active days; (*ii*) *Viewing-specific*: median views per broadcast; (*iii*) *Arrival time-specific*: median arrival time per broadcast (*iv*) *Loyal viewer-specific*: number of unique viewers, number of loyal viewers; (*v*) *Broadcast-specific*: the broadcast generation time of the day, the time elapsed from the last broadcast to the beginning of the predicted one.

We standardize the feature vectors by using the Robust Scaler [7], which removes the median and scales the data according to the quartile range to avoid the influence of outliers. After preprocessing, the NN first encodes 6 groups of closely related features into 6 new features by 6 1-Dimensional Convolutional Neural Networks (1D-CNNs). Then, the new features and the remaining features are input to a fully connected network with 3 hidden layers. Noticeably, in order to meet restrictions 2 and 3, 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}:

$$loss = diff^2 \times (sign(diff) + \alpha)^2 \tag{4}$$

$$diff = Arv_{pred} + t_{relay} - Arv_{actual} \tag{5}$$

where $sign(.)$ is sign function, which returns -1 if the input is negative, 1 otherwise. $\alpha \in (0, 1]$ is used to penalize the overestimation, and we use $\alpha = 0.95$.

We use 5-fold validation to evaluate the model, and plot the resulting distribution of prediction $error = Arv_{actual} - Arv_{pred}$ in Figure 13. As shown in the CDFs, 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 [22], [24], due to the uncertainty of viewers' interests and little to no information is available when the broadcast just initiates. Besides, our intention to avoid overestimate also distorts the model to some degree.

More importantly, our focus is not on a very high prediction accuracy here, but to use the result as a pre-fetching timing reference. With that said, as long as the predicted value is less than the actual value (satisfying restrictions 2 and 3), less wasteful uploads will occur, thus resulting in resource saving, without an impact on viewers. In this respect, our model can meet the restriction conditions in over 99% of all the cases. Finally, due to the help of our system, each broadcast can save 10 seconds of uploading content, on average.

Results: Putting the above designs together, we evaluate the efficacy of the pre-fetching scheme. We use the first 5 days

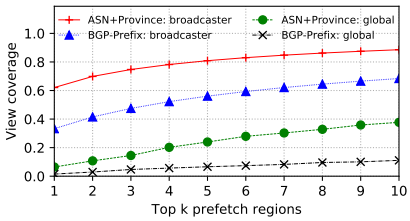


Fig. 14: View coverage of pre-fetch strategies, when pre-fetch at different number of locations.

of viewing records as the training set (to compute the top regions), and the remaining 5 days to test.

We define the metric *view coverage* as the ratio of views localized by pre-fetching, over the number of views for individual broadcasts. The closer the view coverage is to 1, the better the efficacy of pre-fetching. Figure 14 presents the results of the average view coverage for all broadcasts across each top region. 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 requests (§III-C). 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 but not least, the broadcaster-specific server selection strategy clearly outperforms the baseline (pre-fetched contents always go to global top locations), thanks to the broadcaster-specific viewing attraction (§III-C).

We finally empirically evaluate the startup delay improvement attained by pre-fetching. Here, the content will be 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*: $R_i = 1 - \frac{delay_i^s}{delay_i^c}$, 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 find that the startup delay of 92% of the cross ASN+Province views can be improved via pre-fetching. The median R over all network regions is as high as 29.5%.

V. Related Work

MLS User Behavior Inspection: The behavioral patterns of MLS and general live streaming systems have been examined in several works. Raman *et al.* [28] explored the video characteristics, and the social engagement of Facebook Live, and highlighted many unwatched broadcasts. Ma *et al.* [21] investigated Inke Live, and identified differences between MLS and conventional live services. Periscope and Meerkat were examined in [34], [32], [30] from the perspectives of latency and usage patterns. There have been several studies of the Twitch live streaming system [12], [11], [38].

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. (*ii*) We identify localized viewing at the BGP-Prefix level, which underpins our subsequent edge server pre-fetching.

MLS System Optimizations: While we focus on live broadcast uploading, resource wastage in the on-demand video has been examined in [39]. The authors proposed an post-streaming wastage analysis algorithm to achieve the best tradeoff between QoE and resource-saving. The Pre-fetching technique is investigated in [26], where Parate *et al.* designed an app prediction system, predicting which app will be used next and ensuring the freshness of the content. In addition, Li *et al.* [18] enhanced video quality by localizing video delivery through caching. Besides, various video optimizations have been proposed. Ghabashneh *et al.* [14] conducted a measurement study on how the CDN cache will interplay with the viewing experience. Based on their observations, they proposed an ABR algorithm incorporated with CDN awareness. Wang *et al.* [36] 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 viewing QoE. Zhang *et al.* [40] 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.

In contrast to these prior studies, we focus more on optimizing the uploading-side. We propose novel methods to adjust unwatched segments' bitrate to reduce uploading wastage and present a pre-fetching scheme to reduce startup delay.

VI. Conclusion

This paper has used a large-scale dataset to carry out a detailed analysis of MLS user behavior. The main findings include redundant uploading, highly localized view locality, and persistent contribution of loyal viewers. Based on these insights, we propose a decision tree based system to adaptively control the bit rate of the uploaded content that goes unwatched. We also devise a pre-fetching system including a pre-fetching locations prediction module and a DNN-based pre-fetching timing prediction module. We have shown their efficacy in offloading server load, and improving QoE as well. We are working with the team of the examined MLS to incorporate these optimizations into their system. As part of our future work, we intend to evaluate them in the wild.

Acknowledgment

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

REFERENCES

[1] Cisco: 2020 global networking trends report. https://www.cisco.com/c/m/en_us/solutions/enterprise-networks/networking-report.html#, 2020.

[2] Crypto-pan. https://www.cc.gatech.edu/computing/Networking/projects/cryptopan/, 2020.

[3] Facebook live. https://live.fb.com/, 2020.

[4] K-means - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html, 2020.

[5] Pca - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, 2020.

[6] Periscope. https://www.periscope.tv/, 2020.

[7] Robustscaler - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html, 2020.

[8] Standardscaler - sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, 2020.

[9] A. Ahmed, Z. Shafiq, H. Bedi, and A. R. Khakpour. Suffering from buffering? detecting qoe impairments in live video streams. pages 1–10, 2017.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun. Comyco: Quality-aware adaptive video streaming via imitation learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 429–437, 2019.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] W.-Y. Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[21] 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.

[22] 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.

[23] Z. Meng, M. Wang, J. Bai, M. Xu, H. Mao, and H. Hu. Interpreting deep learning-based networking systems. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 154–171, New York, NY, USA, 2020. Association for Computing Machinery.

[24] 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.

[25] 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.

[26] 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.

[27] H. Peng, Y. Zhang, Y. Yang, and J. Yan. A hybrid control scheme for adaptive live streaming. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2627–2631, 2019.

[28] 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.

[29] P. R. Rosenbaum and D. B. Rubin. Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *The American Statistician*, 39(1):33–38, 1985.

[30] 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.

[31] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 272–285, 2016.

[32] 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.

[33] 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.

[34] 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.

[35] C. Wang, J. Guan, T. Feng, N. Zhang, and T. Cao. Bitlat: Bitrate-adaptivity and latency-awareness algorithm for live video streaming. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2642–2646, 2019.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.