Mobile Instant Video Clip Sharing With Screen Scrolling: Measurement and Enhancement

Lei Zhang, *Student Member, IEEE*, Feng Wang, *Member, IEEE*, and Jiangchuan Liu, *Fellow, IEEE*

***Abstract*—Today's multimedia content generation and sharing have been dramatically boosted by the deep penetration of broadband wireless accesses and the much improved processing power of smart mobile terminals. Mobile users can now instantly capture and share short video clips (usually of several seconds) anywhere and anytime, and consume them with convenient touch screen operations. The *instant video clip sharing* has emerged as a mainstream application; such pioneers as Twitter's Vine, Miaopai, Instagram, and Snapchat have seen great acceptance, particularly by the youth community. In this paper, we present an initial study on instant video clip sharing. Taking Twitter's Vine as a representative, we systematically investigate its distinct mobile interface, service framework, and user watching behaviors, revealing how this mainstream multimedia service type differentiates from its traditional counterparts. Our trace measurement and analysis demonstrate that instant mobile video clips have a much shorter lifespan and highly skewed popularity that quickly decays over time. This is further aggravated by the unique screen scrolling operation for video browsing. As such, the download-and-watch scheduling used by existing platforms can hardly achieve quality user experience and cost efficiency. We closely investigate and model the input user gestures for scrolling, including *drag* and *fling*, and analyze the scheduling policy, partitioning it into prefetching scheduling and watch-time download scheduling. We develop effective solutions toward both subproblems as well as their integration with screen scrolling. The superiority of our enhancement is demonstrated by extensive trace-driven evaluation.**

***Index Terms*—Mobile, instant video, efficiency.**

I. Introduction

IN THE past two decades, we have witnessed the great success of multimedia content sharing, in particular online video sharing, and its rapid evolution. The first generation, sharing over the Internet, is represented by a number of video sharing sites (VSSes) such as YouTube [1], [2]. Later, online social networks (OSNs), e.g., Facebook and Twitter, emerged to offer the second generation video sharing, in which users access multimedia content through proactively sharing the video links from external VSSes among friends [3], [4]. Recently, the rapid development and penetration of mobile social networking have enabled the third generation video sharing services that use smart mobile terminals to instantly capture and share ultra-short video clips (usually of several seconds). Many mobile apps, e.g., Twitter's Vine, Instagram, and Snapchat, to name but a few, have incorporated such multimedia services and seen great acceptance, particularly by the youth community [5]. It has also become a mainstream service type in China, where similar emerging apps (e.g., Miaopai, Weishi, Kuaishou, Douyin, Huoshan, etc.) have attracted tremendous amount of users and investments. For example, Miaopai with 70 million daily active users closed a $500 million funding round in 2016,[1] and it now handles 1.5 million uploads per day, with 2.5 billion videos watched every 24 hours; Kuaishou with 50 million daily active users who upload 10 million videos per day, received a $350 million investment from Tencent in 2017.[2] The instant video clips in these services are directly consumed at smart-terminals with specially designed mobile interfaces and operations. The expanded social relations and the distinct operations on the mobile terminals, particularly *screen scrolling*, have greatly increased the amount of videos available to watch, and in the meantime, shorten the time focusing on individual videos from tens of minutes to only a few seconds.

User experience is crucial to mobile instant video clip sharing. An instant video clip itself is of only several seconds long, thereby a mobile user can hardly tolerate a long delay, which would completely ruin the viewing experience. A straightforward solution is to pre-fetch video clips, which is known to be cost-effective and energy-efficient [6]. Yet given the massive ultra-short video clips, deciding which to pre-fetch and when to pre-fetch become much greater challenges. Users of mobile instant video clips also tend to make requests for but fail to finish watching the video clips, many of which even have no chance to start playing with fast screen scrolling. Smart and adaptive watch-time scheduling is thus needed to cope with these distinct operations in the mobile context.

To the best of our knowledge, this new service type has not yet been studied in the literature. In this paper, we present an

Manuscript received May 3, 2017; revised September 15, 2017; accepted December 20, 2017. Date of publication January 17, 2018; date of current version July 17, 2018. This work was supported in part by the Qatar National Research Fund (a member of Qatar Foundation) under NPRP Grant [8-519-1-108] and in part by the Natural Sciences and Engineering Research Council of Canada. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Balakrishnan Prabhakaran. *(Corresponding author: Jiangchuan Liu.)*

L. Zhang and J. Liu are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: lza70@cs.sfu.ca; jcliu@cs.sfu.ca).

F. Wang is with the Department of Computer and Information Science, University of Mississippi, University, MS 38677 USA (e-mail: fwang@cs.olemiss.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2018.2794760

[1] https://www.techinasia.com/china-miaopai-500million-funding-weibo

[2] https://techcrunch.com/2017/03/23/tencent-back-chinese-instagram-kuaishou/

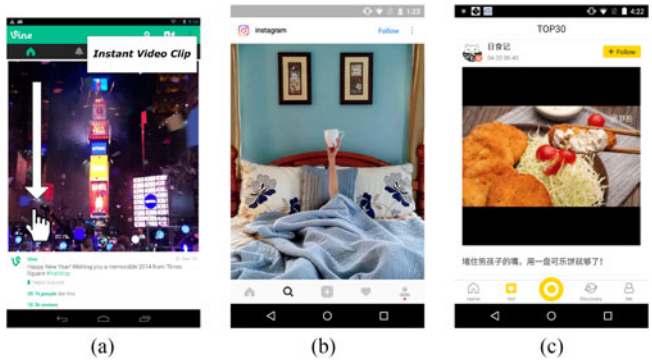


Fig. 1. Typical main interface of mobile instant video sharing (a) Vine. (b) Instagram. (c) Miaopai.

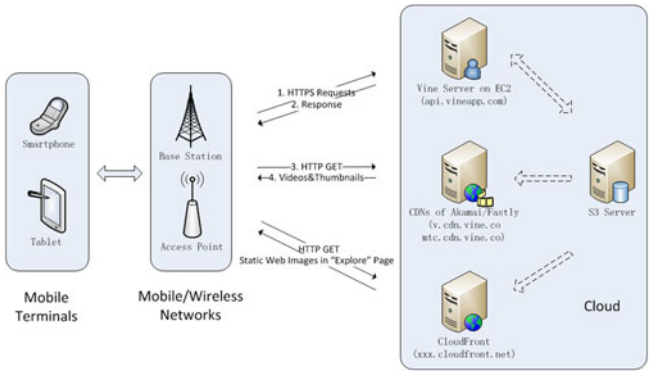


Fig. 2. Service framework of mobile instant video clip sharing (Vine as an example).

initial study on mobile instant video clip sharing empowered by a combination of advanced mobile and cloud platforms. With Twitter's Vine as a representative, we systematically investigate the distinct interface and service framework of this mainstream service type, and identify the unique viewing behaviors, including *batch views* and *passive views*. We develop a data collection engine to track the metadata of video clips and user accesses from Vine. Compared to early-generation videos, the instant mobile video clips have much short lifespan and highly skewed popularity that quickly decays over time, which is aggravated by the unique screen scrolling operation. As such, the download-and-watch scheduling widely used by existing platforms can hardly achieve quality user experience and high cost efficiency. We closely investigate and model the user gestures for scrolling, including *drag* and *fling*, and analyze the scheduling policy, partitioning it into pre-fetching scheduling and watch-time download scheduling. We then develop effective solutions towards both subproblems as well as their integration with screen scrolling. The superiority of our enhancement is demonstrated by extensive trace-driven evaluations.

II. Overview of Mobile Instant Video Clip Sharing

A. Background and Motivation

We next present a case study on Twitter's Vine, which enables users to create ultra-short video clips (limited to a maximum of 6-seconds), as well as post and share them with followers or in OSNs, particularly Twitter (which acquired Vine in October 2012) and Facebook. Vine exclusively focused on mobile users from the very beginning, attracted over 200 million active users since its initial release in January 2013, and continued its core service on Twitter. Other products in the market, e.g., Instagram, Snapchat, Miaopai, and Kuaishou, share similar service architectures and interfaces.

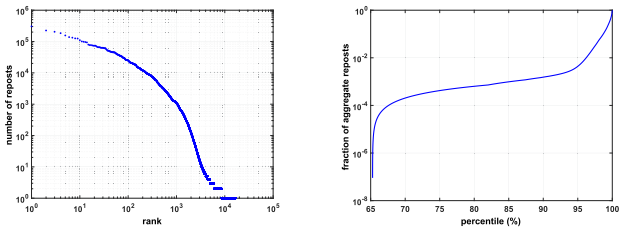
With a Vine client, a user can view, like, comment, and share (repost) the recent posts from others in the `Home/Feed` page, which is, as shown in Fig. 1, a typical and necessary interface for mobile instant video clip sharing and is commonly seen in similar apps. The user can also search for video clips and people of interest, and dedicated channels for specific topics in the `Explore` page. Compared to traditional OSNs with *follower-followee* social relationships, a key (and significant) difference is that the media of interest here are ultra short video clips. This makes its user experience notably different.

B. Service Framework

We have conducted a traffic measurement from our university campus on Twitter's Vine. We captured the traffic between test devices and servers, and intercepted the SSL connections between them to view detailed requests from the application by using the `mitmproxy` tool. The traces show that Twitter builds the mobile instant video sharing system based on a cluster of cloud services, including Amazon EC2, Amazon S3, and Amazon CloudFront, as well as CDNs provided by Akamai and Fastly. We accordingly illustrate the service framework in Fig. 2. A Vine client initiates and maintains a HTTPS connection with the application server running on the EC2 instances with domain name `api.vineapp.com`. After an authorization process, the user can make requests, and the server in turn offers responses for the user to complete such actions as browsing, search, post, comment, and like. When the user logs into the app (or returns to the `Home/Feed` page), the client makes a `GET` request for the timeline information, which corresponds to the recent updates. After receiving the response, the client can further make GET requests to CDNs with domain name `v.cdn.vine.co` or `mtc.cdn.vine.co` to download the video clips and the corresponding thumbnails. From the metadata in captured packets, we infer that the videos and the thumbnails are stored on Amazon S3. Similar operations are performed when visiting the `Explore` page. A slight difference is that the static web images in the page layout are distributed by Amazon CloudFront.

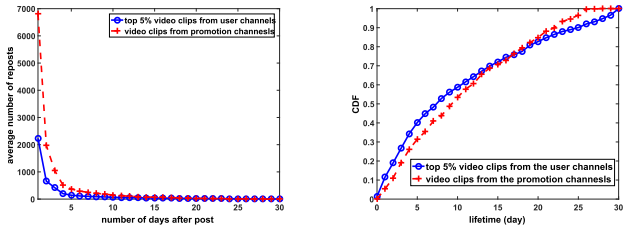
C. Screen Scrolling and Key User Behaviors

In traditional VSSes and OSNs, users need to click to view or link to one specific video, which only allows them to view one video each time/click. Vine-like services, however, return a playlist of video clips when a user touches the screen to view the updates for certain users, tags, or channels. As the user scrolls the smartphone/tablet's screen instant video clips are seamlessly played from the generated list. Scrolling includes



(a) (b)

Fig. 3. Video popularity (a) Video clips rank ordered by the number of repost (b) Skewness of popularity across video clips from the user channels.



(a) (b)

Fig. 4. Video lifespan and propagation. (a) Daily number of reposts. (b) CDF of lifespan.

a series of user gestures, typically *click*, *drag* and *fling*, and the speed, acceleration, and continuity vary depending on the user's input. Given the fixed organization of instant video clips in playlists, it has become an essential user action.

We use *Batch View* to refer to the unique user behavior of viewing multiple video clips with screen scrolling. The batch view implies that mobile users can watch a considerable amount of instant video clips within the playback time of one conventional video (e.g., from YouTube). A related new behavior is *Passive View*. The media contents are arranged in order and a user has limited control over the order for playback (recall the Vine case). For two video clips of interest, if they are separated in the playlist, the user may have to download (and watch) all the video clips between them. These videos of no interest have to be passively watched, and resources for downloading and playing them will be consumed.

III. A Closer Investigation: Measurements and Observations

A. Properties of Instant Video Clips

1) Datasets: We developed customized crawlers and collected the traces of Vine videos that were posted in 16 user channels (47,794 posts) and 2 promotion channels (8,891 posts). For each instant video clip, we accessed and recorded its repost history, including the exact time when it was shared and the user who reposted it. User channels focus on dedicated topics, where each channel has two sections: *recent* and *popular*. An instant video clip can be uploaded to any of the recent sections in these 16 channels, and each user channel lists a small number of popular posts in the popular section. The promotion channels do not accept the posts directly from the normal users; Instead, they choose the most popular and most trending videos clips among all the recent posts in Vine.

2) Popularity: We use the number of reposts to evaluate the video popularity, since the actual number of views for each video clip is hard to obtain by our crawlers. Fig. 3(a) plots the number of reposts as a function of the rank of the video clip by its popularity for all 16 user channels. The plot does not follow a Zipf distribution (which should be a straight line on a log-log scale). This result is different from the previous observations on traditional video sharing services: While the popularity of YouTube videos exhibits a Zipf-like waist with a truncated tail [1], [2], the requests distribution versus video ranks of Renren (the largest Facebook-like service in China) videos follows a perfect power-law pattern [4]. To further understand how the popularity is distributed among Vine videos, we plot the cumulative proportion of the total number of reposts versus the percentile of the video clip in Fig. 3(b). As shown, the popularity of video clips in the user channels is extremely skewed: the top 5% video clips accounts for more than 99% reposts. It heavily deviates from the Pareto Principle (or 80-20 rule). This result is quite surprising, since other video sharing services show much smaller skewness: the top 10% popular YouTube videos account for nearly 80% of views [1]; whereas the top 2% videos in Renren take up 90% of the total requests, and the 5% videos attract 95% of requests [4]. The popularity distributions for different generations of video sharing services show a trend of becoming more and more skewed throughout the 3-stage evolution (YouTube: 10%–80%; Renren: 2%–90%, 5%–95%; Vine: 2%–95%, 5%–99%). The YouTube result implies that, originally, users' interests across videos are not evenly distributed (biased towards popular videos). People tend to watch what others have watched, which is exaggerated when OSNs are introduced, as users in the same social group share common interests. On top of social networking, Vine-like services further offer users ubiquitous mobile accesses, which lead to a more efficient and more extensive propagation of the instant video clips.

3) Lifespan and Propagation: To investigate how the number of reposts changes with time, we plot Fig. 4(a), which shows the average daily number of reposts after the video clips were created. As the popularity of the collected video clips is highly skewed, we only consider popular video clips in the following analysis, specifically, the top 5% reposted video clips from the user channels and all the video clips from the promotion channels. One may notice that the plot lasts slightly longer than the data collection period. This is because many of the video clips that we explored may have been there for a while when we started crawling. In Fig. 4(a), the average number of reposts for the popular video clips monotonically decrease day by day. Even for many of the popular video clips, they are most popular during the first day after the initial posts and are getting less and less popular afterwards. This fast decay feature of mobile instant video clips is quite unique: YouTube videos also reach the global peak immediately after introduction to the system, but decay much more slowly, while the requests for the new videos published in Renren generally experience two or three days latency to reach the peak value, then change dynamically with a series of unpredictable bursts [4].

By defining the *active lifespan* of a video post as the duration from its initial post to the first day in which it gets no repost, we plot the CDF of active lifespan of the popular video clips in Fig. 4(b). Here we use a real value (0) as the threshold to decide whether the video clip is active in propagation, instead of other metrics such as the changing rate and the moving average. The reason is two-fold: first, as shown in Fig. 4(a), although the number of reposts for the popular video clips may change dramatically in the first few days, it still can remain a large value; second, we can hardly know the impact of one repost, as the number of passive viewers after each repost varies significantly (if the user who shares the video clip has a large population of followers, this repost can have a potentially significant impact on the propagation of the video clip). Even with such a "loose" definition of active lifespan, Fig. 4(b) shows that more than half of the popular video reposts can only stay in active for less than 10 days. This result is quite different from the related observations on traditional video sharing services: popular videos in Renren can continuously attract requests for several months [4]; some of YouTube videos can still get views even after 1 year since they were published, which implies that YouTube users' interests are video-age insensitive in a gross scale [1]. The fast decay feature can be possibly explained by the mobile nature of ubiquitousness. As mobile users can upload, and more importantly watch instant video clips at any time and anywhere, they can propagate very efficiently and extensively, and thus can reach the peak immediately. And the frequent video watching and uploading from the mobile user also accelerates the fade of existing instant video clips, even for the popular ones.

B. Summary and Implications

We have revealed a series of unique features of mobile instant video clip sharing, including the extreme skewness, fast decay, and short lifespan. Although lack of common VCR controls (such as rewind and fast forwarding), batch/passive views with scrolling are effective in approaching successive instant video clips in the playlist, enabling users to find interesting contents more easily, and accelerating the propagation of popular videos. Yet, if not being handled properly, screen scrolling may ruin the viewing experience. Currently, most Vine-like services employ a naive download-and-watch scheme, which is clearly not an optimal solution. In the worst case (e.g., downloading every instant video clip through a poor cellular connection), a vicious circle can be formed: the downloading of a just skipped video will take up the network resources and block the downloading of those of interest, which will in turn force the user to give up watching the target videos and scroll forward to search for other interesting videos.

As the unpopular video clips move towards the bottom of the playlist, users hardly see them again. On the contrary, the popular video clips will be promoted to the popular section, and users can easily reach these posts. They become more and more popular, keeping on the top of the playlist and thereby being accessed more frequently. With the batch and passive views, the above process is accelerated and exacerbated. Although this extreme skewness suggests that identifying the popular videos and pre-fetching them could be beneficial, deciding which video clips will fall into the user's interests is never an easy task. Moreover, the short lifespan and fast decay imply the popular contents are much more dynamic than those in other mobile VoD or video streaming applications. This introduces a dilemma for pre-fetching: on one hand, we would like to cache as many videos as possible to provide fluent watching experience; on the other hand, if the cached videos cannot be watched soon enough by the user, it becomes a huge waste for fetching them, as they will probably be flushed out by more recent feeds, having no chance to be viewed. As such, neither a simple download-and-watch scheme nor a naive pre-fetching/caching scheme would work efficiently, and a smart adaptive solution is expected. More importantly, it must work well with screen scrolling, a rich operation whose multiple factors, e.g., speed/acceleration, are to be considered.

IV. Enhancements on Mobile Side: Pre-Fetching and Watch-Time Scheduling

A. Problem Formulation

We now present a generic formulation for the video download scheduling problem in mobile instant video sharing. As mentioned, instant video clips are usually organized in different playlists, which can be characterized into three types: the list of video updates from followees (social videos), the list of promoted videos in popular sections (popular videos), and the list of user uploaded videos in recent sections (recent videos). Only the playlist of social videos changes with different users, and the other two types of playlists remain the same across users. Consider one video watching event of a specific user. Denote the playlist of instant video clips that will be watched by the user as $V = \{v_1, v_2, ..., v_n\}$. As illustrated in Fig. 5, according to different user input actions, each instant video clip may remain in the user's viewport for for a specific duration. We use $U = \{u_1, u_2, ..., u_n\}$ to denote such durations, where u_i corresponds to the duration that the user watches video v_i. Also, we let u_0 denote the time that the user starts watching the playlist. We consider two types of network connections in this formulation: mobile cellular connections (e.g, 3/4G) and wireless local connections (e.g., WiFi). We use $B(t)$, $C(t)$, and $E(t)$ to denote the available bandwidth, the monetary cost, and the energy consumption at a given time t, respectively, where $B(t) \in \{B_{\text{wifi}}, B_{3/4G}\}$, $C(t) \in \{C_{\text{wifi}}, C_{3/4G}\}$ ($C_{\text{wifi}} = 0$, since the cost for WiFi connections is usually negligible), and $E(t) \in \{E_{\text{wifi}}, E_{3/4G}\}$. As in previous studies [7], [8], we divide the time evenly into discrete time slots. Let R_i be the video streaming rate of video v_i, and L be the maximum video length. In practice, most users capture video clips till reaching the maximum length (in Vine's case, 6 seconds); hence, their file sizes after transcoding to a certain resolution are almost the same, i.e., R_i and L can be treated as given constants.[3] Define a video downloading

[3]It is worth noting that for ease of exposition, here we assume homogeneous video length. Our model and solutions can be easily extended to afford various specifications for individual video, which does not change the fundamental problem studied in this paper.

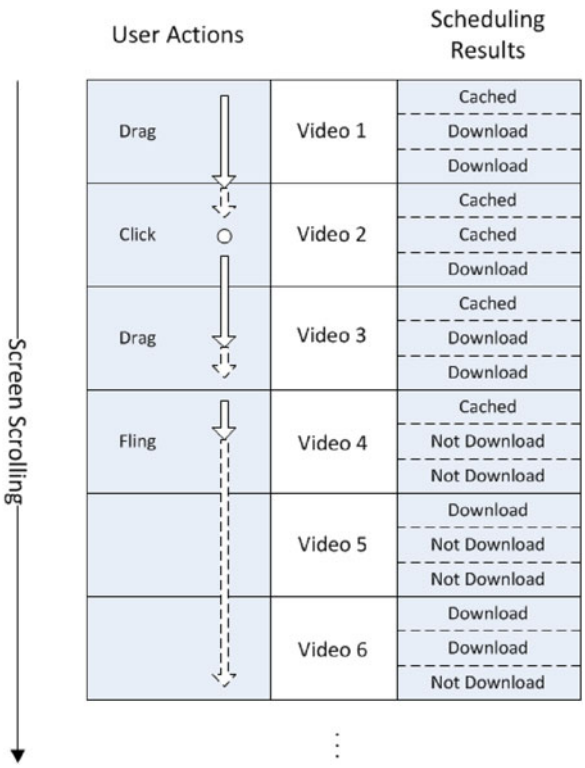


Fig. 5. An illustration of playlist optimization.

schedule as $S = \{(\hat{v_1}, \hat{t_1}, \hat{l_1}), (\hat{v_2}, \hat{t_2}, \hat{l_2}), ..., (\hat{v_k}, \hat{t_k}, \hat{l_k})\}$, where a tuple $(\hat{v_i}, \hat{t_i}, \hat{l_i})$ ($\hat{v_i} \in V$ and $\hat{l_i} > 0$) means at time $\hat{t_i}$, we start to download video $\hat{v_i}$ for the duration $\hat{l_i}$.

Our problem is to find a proper schedule S that optimizes the video watching experience with high efficiency in terms of monetary cost and energy consumption. We define the playback discontinuity of a single video v_i watched for the duration u_i as:

$$\text{discontinuity}(v_i) = 1 - \frac{1}{\min(u_i, L)} \cdot \sum_{t \in \left(\sum_{k=0}^{i-1} u_k, \min\left(\sum_{k=0}^{i} u_k, \sum_{k=0}^{i-1} u_k + L \right) \right]} \mathbb{I}\left[\sum_{\substack{(\hat{v}_j, \hat{t}_j, \hat{l}_j) \in S, \\ \hat{v}_j = v_i, \hat{t}_j \le t}} \sum_{\hat{t}=\hat{t}_j}^{\min(\hat{t}_j + \hat{l}_j - 1, t)} B(\hat{t}) \ge \left(t - \sum_{k=0}^{i-1} u_k \right) \cdot R_i \right],$$

where $\mathbb{I}[\cdot]$ is the indicator function. Inside the indicator function, it checks whether the download progress stays ahead of the playback at any given time slot. Given the watch duration (if it is longer than the video length, we use the video length instead), we can calculate the ratio of continuous playback, and thus define the playback discontinuity accordingly (between 0 and 1). The single video playback discontinuity naturally reflects the user experience for a continuous playback, which calculates how many time slots the downloading of this video misses the deadline for the playback. We further define the playback discontinuity of the playlist V as a weighted sum of those of individual videos:

$$\text{Discontinuity} = \sum_{v_i \in V} w_i \cdot \text{discontinuity}(v_i),$$

where w_i is the normalized weight for v_i. An intuitive assignment of w_i can be $\frac{1}{\sum_{k=1}^{n} u_k} u_i$, which assigns higher weights to the videos that have longer watching durations, as longer watching durations usually imply higher user interests. We will further discuss more specific assignments of w_i later.

Our objective is thus to minimize the playback discontinuity, as well as the total monetary and energy cost:

$$C_{\text{total}} = \sum_{(\hat{v}_j, \hat{t}_j, \hat{l}_j) \in S} \sum_{t=\hat{t}_j}^{\hat{t}_j + \hat{l}_j - 1} C(t),$$

$$E_{\text{total}} = \sum_{(\hat{v}_j, \hat{t}_j, \hat{l}_j) \in S} \sum_{t=\hat{t}_j}^{\hat{t}_j + \hat{l}_j - 1} E(t).$$

It is easy to see that the above objectives contradict with each other, as downloading more portions of the playlist can reduce the playback discontinuity but will also inevitably consume more energy and may increase the monetary expense. We thus adopt the following linear combination form to align them together:

$$p \cdot \text{Discontinuity} + q \cdot \frac{C_{\text{total}}}{C_{\max}} + r \cdot \frac{E_{\text{total}}}{E_{\max}}, \tag{1}$$

where p, q and r are the parameters to assign different weights to the three goals. As Discontinuity is a ratio between $[0, 1]$, we also normalize the monetary cost and the energy consumption by their corresponding maximum values, where $C_{\max}$ is the maximum total cost of the case that all the videos in the playlist are downloaded through 3/4G links, and $E_{\max}$ can be obtained similarly. We then have the following theorem:

Theorem 1: The decision version of the modeled generic downloading scheduling problem is NP-complete.

Proof: The key for the proof is to show that the Knapsack problem can be reduced to the simplified version of our problem. See the detailed proof in Appendix A. ■

B. Pre-Fetching Scheduling

We first consider pre-fetching, which happens well before the user starts watching the playlist, i.e., without a stringent time constraint; hence we can offload the mobile traffic to the wireless network to reduce the transmission cost. The objective is to find a schedule S_{pf} to pre-fetch the videos, subjecting to the following constraints:

1) Storage Constraint:

$$\sum_{(\hat{v}_j, \hat{t}_j, \hat{l}_j) \in S_{pf}} \sum_{t=\hat{t}_j}^{\hat{t}_j + \hat{l}_j - 1} B(t) \le \text{StorageSize};$$

2) Cost Constraint:

$$\forall(\hat{v}_j, \hat{t}_j, \hat{l}_j) \in S_{pf}, \sum_{t=\hat{t}_j}^{\hat{t}_j+\hat{l}_j-1} C(t) = 0.$$

The storage constraint ensures that the total amount of pre-fetched video will not exceed the limited local storage. And the cost constraint implies that the pre-fetch is performed only through WiFi links. As the pre-fetched videos may not be watched during the watch-time, the performance gain of pre-fetching is uncertain. In order to achieve the unguaranteed performance gain with lower costs, we do not consider cellular communications during pre-fetching. The playlist V during the pre-fetching may only be a subset of that during the watch-time, as pre-fetching occurs before video watching and new videos may be added to the playlist after pre-fetching, which will be handled by the watch-time download scheduling to be discussed lately.

Given that the user behavior during the video watching is unknown at this stage (nor the watching duration U), we thus introduce $P = \{p_1, p_2, ..., p_n\}$ to denote the user preference on each video in the playlist, which can reflect the potential lengths of the watch durations. In practice, P can be evaluated by video popularity, video timeliness, or the social distance between the publisher (the user who reposts the video) and the consumer (the user who may watch the video), or a combination of them. Without loss of generality, here we use the video popularity as the metric of user preference. In addition, we introduce parameter $\alpha \in [0, 1]$ to represent the aggressiveness of the pre-fetching. For each instant video clip, we pre-fetch α of the total video, instead of downloading the whole clip. The playback discontinuity of a single video v_i can then be rewritten as

$$\text{discontinuity}(v_i) = 1 - \frac{1}{\alpha \cdot L} \cdot \frac{pf(v_i)}{R_i},$$

where $pf(v_i)$ defines how much of v_i has been pre-fetched:

$$pf(v_i) = \sum_{(\hat{v}_j, \hat{t}_j, \hat{l}_j) \in S_{pf}, \hat{v}_j = v_i} \sum_{\hat{t}=\hat{t}_j}^{\hat{t}_j+\hat{l}_j-1} B(\hat{t}).$$

The next step is to find a proper assignment of w_i. For this subproblem, we define w_i as

$$w_i = \frac{1}{\sum_{k=1}^{n} p_k \cdot \text{discontinuity}(v_k)} p_i \cdot \text{discontinuity}(v_i),$$

which considers both the user preference for v_i and its current playback discontinuity. Note that w_i decreases as more of v_i has been pre-fetched, as given the batch view behavior, it is not reasonable to allocate all the resources to a tiny portion of extremely popular videos. In practice, the first several units of a video are normally requested with a much higher probability than its later part. Together with the pre-fetching aggressiveness α, this assignment of w_i allows us to pre-fetch more videos with the video preference still being considered.

As the monetary cost for WiFi links is usually negligible, our goal here is to minimize Discontinuity with the form:

$$\text{Discontinuity} = \frac{1}{\sum_{k=1}^{n} p_k \cdot \text{discontinuity}(v_k)} \cdot \sum_{v_i \in V} p_i \cdot \text{discontinuity}(v_i)^2.$$

Different from (1), this objective function does not directly involve the energy consumption of pre-fetching. Here, we use α to control the trade-off between the energy consumption and the playlist playback discontinuity. As α gets larger, more videos would be pre-fetched, which consumes more energy; on the contrary, if α is small, only a small portion of videos will be pre-fetched, and thus little energy is consumed. Therefore, the above objective function can still represent the overall performance.

This pre-fetching scheduling subproblem is a variation of the knapsack problem with a total weight limit:

$$W = \min(\text{StorageSize}, \sum_{\forall t \text{ such that } C(t)=0} B(t)),$$

where an object is one time slot length of video playback, and its value is the amount of decrease of $p_i \cdot \text{discontinuity}(v_i)^2$ after pre-fetching one more time slot, if the object belongs to video v_i. It is easy to see that while the weight of each object is the same, the value changes as the decisions are made, i.e., as one object of video v_i is downloaded, the value of all the remained objects of video v_i decreases as now $\text{discontinuity}(v_i)$ decreases. We use a greedy algorithm to search and download one object that currently has the greatest value in each iteration. Recall that all the objects have the same weight. Given the optimal result in each iteration, the algorithm returns the final optimal pre-fetching schedule.

C. Watch-Time Download Scheduling

Unlike pre-fetching, the video watching durations can be largely determined from the input user gestures, typically *click*, *drag* and *fling*, where the last two gestures can cause screen scrolling. Once a gesture is given, the following process of screen moving is predetermined. Given the fixed display size of each clip (specifically, the fixed height), the motion of screen scrolling can be modeled and calculated, and the details of the scrolling process can be obtained accurately (e.g., how many videos are present, how long each video will stay in the viewport), which can hardly be done in VoD or video streaming applications. Although different operating systems have different technical details for implementation, the philosophy for animating the screen scrolling is generally the same, which is to gradually decelerate the scrolling speed until it reaches zero if there is no other finger touch detected during the deceleration.[4] We next show how to calculate the video watching durations from the input user gestures, by taking the Android OS as an example.

By detecting and collecting information about the user's finger touch, the initial scrolling speed s_0 can be calculated as

[4]https://developer.android.com/training/gestures/scroll.html

the dragging distance divided by the touch time in the unit of pixels/second. As the screen only scrolls vertically in mobile video sharing services, let h denote the height of each instant video clip. By default, the threshold s_T for the initial scrolling speed to distinguish between a drag and a fling in the Android OS is 50 pixels/second, which can be scaled under different configurations based on the actual screen resolution.

Dragging: In the case of dragging, the screen scrolling speed will experience a uniform deceleration with the default deceleration $d = 2000$ pixels/second^2. Given the initial speed s_0, there will be $\lfloor s_0^2/2hd \rfloor$ video clips covered by this drag gesture. In the deceleration process, for the m-th video clip showing in the dragging animation, we have

$$mh = s_0 t_m - dt_m^2/2, \tag{2}$$

where t_m is the time that the $(m+1)$-th video clip starts to enter the viewport ($t_0 = 0$). Solving (2) gives us

$$t_m = (s_0 - \sqrt{s_0^2 - 2mhd})/d \ (1 \le m \le \lfloor s_0^2/2hd \rfloor). \tag{3}$$

Flinging: If a fling is detected, the deceleration will change with the scrolling speed. Given the scrolling speed s, the total fling duration $T(s)$ and the total fling distance $D(s)$ can be calculated by using the following equations:

$$l(s) = \log\left[0.35 \cdot s/(\text{Fric} \cdot P_{\text{COEF}})\right], \tag{4}$$

$$T(s) = 1000 \cdot \exp[l(s)/(D_{\text{RATE}} - 1)], \tag{5}$$

$$D(s) = \text{Fric} \cdot P_{\text{COEF}} \cdot \exp[D_{\text{RATE}}/(D_{\text{RATE}} - 1) \cdot l(s)], \tag{6}$$

where $D_{\text{RATE}} = \log(0.78)/\log(0.9)$, Fric denotes the parameter of the friction with the default value as 0.015, and $P_{\text{COEF}} = G \cdot 39.37 \cdot ppi \cdot f_c$. To compute P_{COEF}, G is the gravity of the Earth with a constant value of 9.80665 m/s^2, 39.37 is used for the conversion between meters and inches, ppi denotes the parameter of pixels per inch for the specific mobile device, and f_c is a user-defined value ($f_c = 0.84$ by default). Instead of using real-world physics, screen scrolling is animated based on platform-standard virtual physics (friction, velocity, etc.). Such parameters as Fric and P_{COEF} that define the virtual physics are adjustable for different devices and applications by users or developers.

From (5) and (6), we can derive

$$D(s) = \text{Fric} \cdot P_{\text{COEF}} \cdot (T(s)/1000)^{D_{\text{RATE}}}. \tag{7}$$

Given the initial speed s_0, there will be $\lfloor D(s_0)/h \rfloor$ video clips covered by this fling gesture. Assume s_m is the scrolling speed at time t_m. In the deceleration process, the following equation is also satisfied:

$$D(s_0) - D(s_m) = mh \ (1 \le m \le \lfloor D(s_0)/h \rfloor). \tag{8}$$

By combining (7) and (8), we have

$$t_m = T(s_0) - T(s_m) = T(s_0) - 1000 \cdot [(T(s_0)/1000)^{D_{\text{RATE}}} - mh/(\text{Fric} \cdot P_{\text{COEF}})]^{\frac{1}{D_{\text{RATE}}}} \ (1 \le m \le \lfloor D(s_0)/h \rfloor). \tag{9}$$

As the basis of this analysis, (2), (4), (5) and (6) are obtained from our analysis of the Android OS source code.[5,6] Assume that the user will focus on one video at any given time. From (3) and (9), the watching duration of m-th video clip showing in the screen scrolling animation can be obtained as $u_m = t_m - t_{m-1}$. Based on the above analysis, we can now tell how many videos are scrolled by a user gesture and how long each video can stay in the viewport. Therefore, the video watching duration $U = \{u_1, u_2, ..., u_n\}$ is available once the input user gestures are given. This subproblem of watch-time download scheduling is thus to find a proper real-time download schedule S_{rd}, so as to minimize our objective:

$$p \cdot \text{Discontinuity} + q \cdot \frac{C_{\text{total}}}{C_{\max}} + r \cdot \frac{E_{\text{total}}}{E_{\max}}.$$

Note that Discontinuity here needs to consider the result of the pre-fetching schedule with the updated discontinuity(v_i):

$$\text{discontinuity}(v_i) = 1 - \frac{1}{\min(u_i, L)} \cdot \sum_{t \in \left(\sum_{k=0}^{i-1} u_k, \min\left(\sum_{k=0}^{i} u_k, \sum_{k=0}^{i-1} u_k + L \right) \right]} \mathbb{I}\left[\sum_{\substack{(\hat{v}_j, \hat{t}_j, \hat{l}_j) \in S_{rd}, \\ \hat{v}_j = v_i, \hat{t}_j \le t}} \sum_{\hat{t}=\hat{t}_j}^{\min(\hat{t}_j + \hat{l}_j - 1, t)} B(\hat{t}) + pf(v_i) \ge \left(t - \sum_{k=0}^{i-1} u_k \right) \cdot R_i \right],$$

where the amount of the video v_i that has been pre-fetched ($pf(v_i)$) is also considered in the calculation. For the watch-time downloading subproblem, we define w_i as

$$w_i = \frac{1}{\sum_{k=1}^{n} u_k^2} u_i^2,$$

which emphasizes the importance of watch durations.

The problem is essentially to trade off between playback discontinuity, monetary cost and energy consumption during the watch-time. Since videos with higher watching durations have higher impacts on the objective function, they should be scheduled for downloading with higher priorities. A heuristic therefore tries to download one time slot of the video v_i that has the highest value of $u_i^2 \cdot \text{discontinuity}(v_i)$ in the unscheduled set, and schedules its downloading interval as late as possible (i.e., closest to, but before its playback deadline) so as to only introduce the minimal impact on other videos to be scheduled later. If there still exist WiFi slots after the initial scheduling finishes, we reschedule the video downloadings that are originally scheduled in the later cellular slots to fill up these WiFi slots.

Algorithm 1 integrates the solutions for both subproblems. V_p is the set of videos that are considered for pre-fetching, which can be initialized as the whole playlist. V_w is the set of videos

[5]https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/widget/Scroller.java

[6]https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/view/ViewConfiguration.java

Algorithm 1: Playlist Scheduling

```
1: while true do
2:   if a video v is shown on screen then
3:     Remove v from V_p;
4:     if a new user input gesture comes then
5:       Set all slots in Q to empty;
6:       Obtain V_w based on the user input gesture
         model;
7:       Sort V_w in descendant order according to
         u_i^2 · discontinuity(v_i);
8:       Delete the videos that cannot reduce the
         objective value if downloaded from V_w;
9:       while V_w is not empty do
10:        Pick the 1st video v_1 out of V_w;
11:        while v_1 is not fully downloaded do
12:          if there exist available slots in Q then
13:            Update Q to assign the
               closest-to-deadline slot to v_1;
14:          else
15:            break;
16:          end if
17:        end while
18:      end while
19:      if there are available WiFi slots in Q then
20:        Update Q to move later scheduled
           downloadings forward to fill the WiFi slots;
21:      end if
22:    end if
23:  else
24:    Update V_p for newly arrived videos;
25:    if cache is not full and WiFi is available then
26:      Search V_p to find the video v with the largest
         decrease amount of p_i · discontinuity(v_i)^2
         assuming one more unit of the video
         is pre-fetched;
27:      Update Q to schedule downloading one unit
         of v;
28:    end if
29:  end if
30:  Download one unit of video if currently scheduled in
     Q and update Q accordingly;
31: end while
```

that are scheduled during watch-time according to the dragging/flinging model given an user gesture input. Q is the schedule queue, which denotes whether each time slot is available or assigned to download one unit of a certain video. The first part (line 2–22) sorts V_w accordingly ($O(|V_w|\log|V_w|)$), and then searches Q for a proper schedule for each video ($O(|V_w||Q|)$). Although this part is executed each time a user gesture is detected, given that any user gesture can only affect a limited number of videos that will show on the screen, $|V_w|$ is quite small, and so is the searching space in Q. In the second part (line 23–28), the searching process will be performed at most $|M||V_p|$ rounds assuming each video is divided into M units. As each round of the search process only has the complexity of $O(|V_p|)$, its efficiency is also acceptable especially given that the pre-fetching happens well before the video watching and is often with a much longer time span.

V. Performance Evaluation

We evaluate our solution using real-world data of Vine videos and the user gestures recorded. For comparison, we implement another two downloading schemes [7]. *Sequential Downloading* (SeqD) downloads all the videos according to their order in the playlist, and disregards all the user actions, which takes the playlist as a single long video, and is the most aggressive downloading scheme with the least flexibility. *Next-one Downloading* (NextD) always attempts to download the next video that enters the viewport, which emulates the caching strategy mostly used in Vine-like services. If the current watching duration is not long enough to download the next video, its playback will be interrupted. Besides the two downloading scheme, we also implement a popularity-based *raw Pre-fetching* (rPF) method, which caches instant video clips solely based on their popularity.

We consider a set of metrics including playback discontinuity, monetary cost, energy consumption in our evaluation, which directly relate to the three goals of our optimization problem. We randomly introduce 20 video watching events from 9 a.m. to 9 p.m. in the week-long dataset, emulate the user behaviors by using the user gesture traces, and produce the average results. Each video watching event consumes 50 videos from popular sections and 150 videos from recent sections. We assume that, during the watch-time only cellular links are available with 1 MB/s bandwidth, and WiFi is available for pre-fetching once every hour. We adopted this setting to emulate a worst case scenario, since if WiFi is available during watch time, our solution can perform better. We assume each Wi-Fi session is long enough to complete a round of pre-fetching. The application local storage is set as 100 MB. The monetary cost model is 10 dollar per 100 MB traffic, which is close to the major mobile operators' data add-on prices.[7] The energy model is adopted from [9], which considers practical details such as the tail time in 3G/4G communications [10].

A. Data Traces of Vine Videos and User Gestures

In our simulations, we use the dataset collected and presented in Section III. Fig. 6 plots the video popularity versus its rank for the two types of videos (from popular sections and recent sections, respectively), in which videos from recent sections exhibit a higher skewness. The average numbers of hourly video uploads during a day for popular and recent sections are shown in Fig. 7.

To obtain the real-world user gesture traces, we implement an Android app to record user touch events. We recruit 10 volunteers to watch Vine videos using the official client on Android under cellular connections and WiFi connections, respectively. Each experiment is conducted around 5 minutes. We plot two important characteristics of user behavior from the collected

[7]http://www.telus.com/en/ca/mobility/prepaid/add-ons/

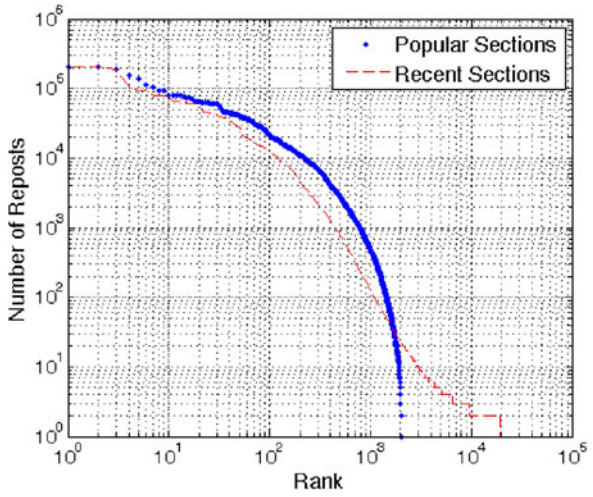


Fig. 6. Popularity distribution.

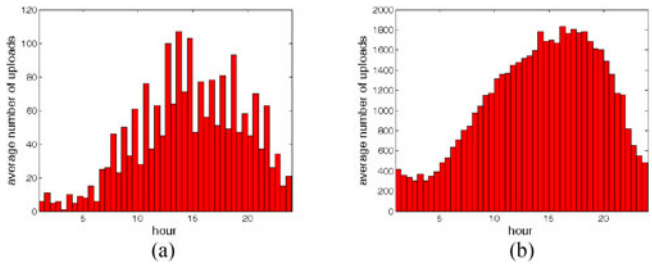


Fig. 7. Average video uploading rate. (a) Videos uploaded into popular sections. (b) Videos uploaded into recent sections.

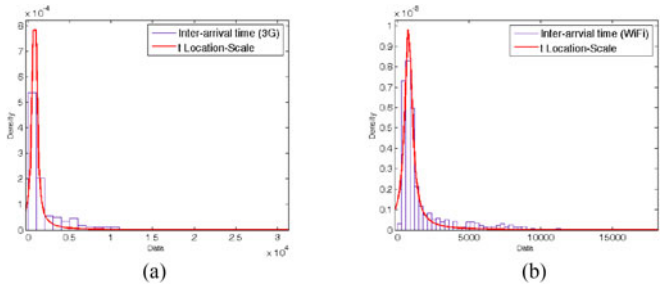


Fig. 8. Density of inter-arrival time of user gestures. (a) Inter-arrival time of user actions under 3G scenarios. (b) Inter-arrival time of user actions under WiFi scenarios.

traces: Fig. 8 shows the probability density distribution of the user gestures' inter-arrival time, and its curve fitting result; and Fig. 9 plots the histogram of the measured initial scrolling speed of the triggered flings. The user gesture traces are used to simulate video watching events by applying the dragging/flinging model.

B. Impacts of p/q and p/r Ratios

As we have three goals in our objective function, two of which are designed for efficiency, we vary the ratio of p/q with $r = 0$, and vary the ratio p/r with $q = 0$, respectively. The results are shown in Fig. 10(a) and (b), which demonstrate how the playback discontinuity, the cost efficiency and the energy

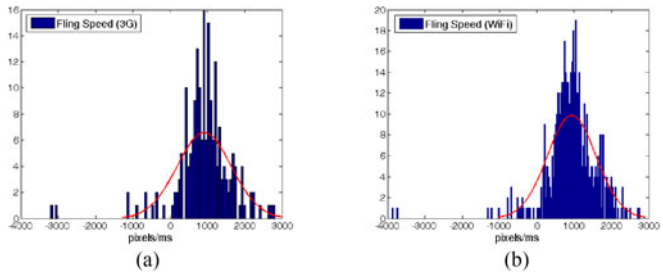


Fig. 9. The initial scrolling speed of flings. (a) Fling speed under 3G scenarios. (b) Fling speed under WiFi scenarios.

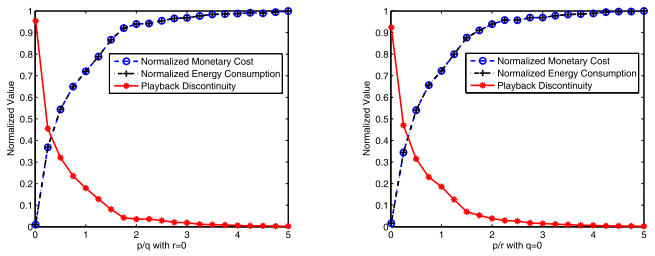
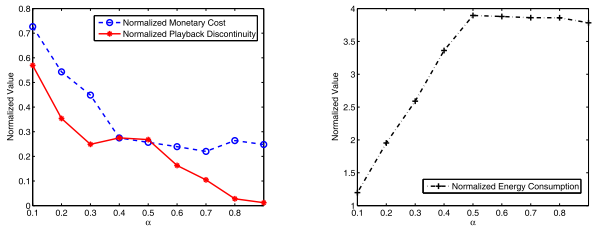


Fig. 10. Impacts of p, q and r. (a) Impact of different p/q values on efficiency and playback discontinuity ($r = 0$) (b) Impact of different p/r values on efficiency and playback discontinuity ($q = 0$).

efficiency change with different parameters. As p, q and r only affect our optimization goal during the watch-time, we disable pre-fetching in this experiment, and only focus on studying the proper settings of p/q and p/r for watch-time downloading (referred to as WT). Since changing the ratios between p, q and r is essentially the trade-off between efficiency and performance, it is not surprising that the plots in Fig. 10(a) and (b) are very similar. Note that, as we adopt both a linear monetary cost model and a linear energy model, the normalized results for monetary cost and energy consumption is nearly the same when pre-fetching has been disabled. When p/q (p/r) is small (<0.5), very limited number of videos have been downloaded and the video playback is severely affected, so as to reduce the monetary cost (energy consumption). On the other hand, if p/q (p/r) increases to a certain degree (>3), the video watching experience is optimized but results in a much higher monetary cost (energy consumption). Moreover, there is a small interval near 1.5 in the figure, in which the playback quality is acceptable with a relatively good cost efficiency (energy efficiency). We thus pick $p/q = 1.5$ and $p/r = 1.5$ as the default setting for the remaining evaluation.

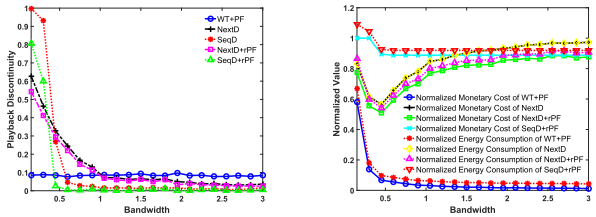
C. Impact of Pre-Fetching Aggressiveness α

We vary α from 0.1 to 0.9, which represents how aggressively our pre-fetching (referred to as PF) acts, and normalize the performance metrics of WT+PF by WT as the baseline. As shown in Fig. 11(a), when α grows large, the playback discontinuity decreases gradually, while the monetary cost first decreases and then becomes stable. This is because when the pre-fetching becomes more aggressive, more videos to be watched are



(a) (b)

Fig. 11. Impact of α (a) Relative monetary cost efficiency and playback discontinuity of WT+PF normalized by WT (b) Relative energy efficiency of WT+PF normalized by WT.



(a) (b)

Fig. 12. Impact of downloading bandwidth when $p/q = 1.5$, $p/r = 1.5$. (a) Playback discontinuity. (b) Monetary and energy costs normalized by the baseline results of SeqD.

downloaded through WiFi, thereby saving the monetary cost; however, as one cannot accurately predict which video (and which portion of it) will be consumed and the application local storage is also limited, there are still some parts of videos to be downloaded. At the same time, as more and more videos are pre-fetched, some of the videos, which originally will not be downloaded during watch-time according to the performance-efficiency trade-off, are pre-fetched and make the playback discontinuity continue to decrease. On the other hand, Fig. 11(b) shows that the normalized energy consumption keeps growing (from 1.2x to 3.8x) until α reaches 0.5, where the application local storage is used up and limits the amount of pre-fetched videos even if α gets larger than 0.5 (and thus the energy consumption). It is worth noting that here the comparison is only between WT+PF and WT, not WT+PF and SeqD (WT+PF and NextD), where, as will be discussed in the next section, the energy saving can achieve as much as over 90%. Fig. 11 suggests us setting $\alpha = 0.2$ as the default value, where WT+PF consumes about 2x energy of WT to save around 45% monetary cost and improve the playback discontinuity by over 60%.

D. Performance Enhancement

We next show the overall performance gain of our solution (WT+PF) compared to Sequential Downloading (SeqD), Next-one Downloading (NextD), and the two downloading schemes with raw pre-fetching (SeqD+rPF and NextD+rPF) under different operating conditions. To conduct a fair comparison, we ensure that PF and rPF consume the same amount of storage. Note that the scheduling results of our pre-fetching (PF) affects directly those of watch-time downloading (WT). Therefore, we consider our solution as a whole piece and compare it with other baseline approaches. To simulate different levels of connection quality, we vary the downloading bandwidth from 150 KB/s to 3 MB/s.

Fig. 12 shows the results with $p/q = 1.5$ and $p/r = 1.5$, which plots the playback discontinuity in Fig. 12(a), and the normalized monetary cost and the normalized energy consumption in Fig. 12(b). As we keep α fixed in this experiment, the monetary cost and the energy consumption exhibit similar patterns due to their linear models, and thus we see overlapping plots for NextD in Fig. 12(b). As SeqD naively downloads videos and disregards their playbacks, which introduces the highest monetary cost/energy consumption, we use it as the baseline, and normalize the costs of the other two approaches. Our proposed approach (WT+PF) shows a very stable performance with different downloading bandwidths, in terms of both playback discontinuity and cost efficiency. On the contrary, the playback discontinuity of NextD and that of SeqD increase dramatically when the bandwidth is low (<0.5 MB/s). The reason is that, as the bandwidth becomes lower, it is more and more difficult for these two downloading schemes to finish each downloading before the playback, while our proposed approach can still keep the playback discontinuity at a low level by smartly managing the downloading according to user actions and efficient pre-fetching. On the other hand, the monetary cost/energy consumption of NextD quickly increases after reaching the minimal at 0.5 MB/s. The reason for both NextD and SeqD introducing high monetary cost/energy consumption is that, when the bandwidth is low, both NextD and SeqD keep the downloading link busy almost all the time; whereas both of the downloading schemes attempt to download all the videos if the bandwidth becomes high enough. After introducing the popularity-based raw pre-fetching, both NextD+rPF and SeqD+rPF provide better viewing experience (lower playback discontinuity) than their counterparts. On the cost efficiency, NextD+rPF causes slightly higher costs than NextD, while SeqD+rPF can save around 10% costs than SeqD when the bandwidth is high. This is because unlike NextD, SeqD can always benefit from pre-fetching. Fig. 12(b) shows that our approach can save at least over 40% monetary cost and 30% energy consumption, and the cost/energy saving under high bandwidths can be higher than 90%.

The other four caching schemes may achieve the similar performance to ours in terms of playback discontinuity when the bandwidth is high enough. This is because the setting of p/q and p/r asks for a balance between cost efficiency and playback discontinuity. If a user is more aggressive on the video watching experience, s/he can further increase of the ratio of p/q and p/r, e.g., to $p/q = 3.5$ and $p/r = 3.5$. We plot the corresponding results in Fig. 13, which shows that our solution can always achieve the best playback discontinuity at diverse bandwidths, and still with huge amounts of (over 90%) cost/energy savings.

VI. Further Discussion

Considerable research efforts have focused on evaluating and improving video services for mobile users over various communication networks [8], [11]–[13], mainstream platforms [14], efficient coding schemes [15], [16], emerging cloud computing

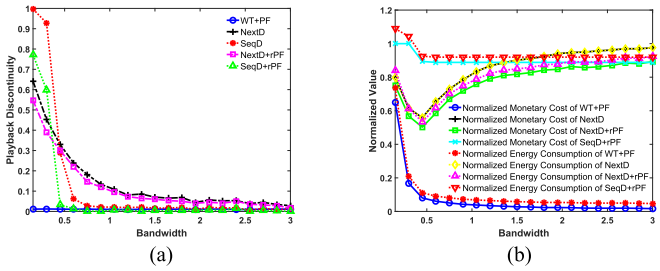


Fig. 13. Impact of downloading bandwidth when $p/q = 3.5$, $p/r = 3.5$. (a) Playback discontinuity. (b) Monetary and energy costs normalized by the baseline results of SeqD.

architecture [17]–[19], and novel transmission standards [20], [21]. Unlike most of the existing studies that either focus on the 1st or the 2nd generation of video sharing service, we for the first time presented an in-depth measurement on mobile instant video clip sharing and took a different approach to investigate the enhancement of this mainstream service type from the mobile perspective.

The increasing availability of sensors integrated in mobile devices further provides ample opportunities for understanding the operating contexts and the surrounding environment. Existing works have been done on energy-efficient mobile context sensing [22], [23]. Further integrating such sensing abilities with a human mobility model based on periodic travels [24] can bring us the chances to make our proposed approach even smarter, i.e, automatically adjusting parameter values of p, q and r according to the currently obtained operating context.

Moreover, extracting patterns from mobile users' daily activities can assist us to find proper locations for pre-fetching [25]–[27]. Mobile users visit certain locations as daily routines (e.g., office, classroom, home), which normally provide dedicated WiFi connections and thus can act as regular pre-fetching sites. The knowledge of the locations with regular user accesses can be more useful as Vine-like services such as Miaopai and Kuaishou allow users to check uploads from nearby users.

Finally, our work has also touched the interests of some other research fields such as popularity prediction and human-computer interaction. Numerous popularity prediction schemes for online contents have been proposed in the literature, most of which focus on predicting the trend based on time series with regression models [28], [29] and classification models [30], [31]. Better video popularity prediction [32] can help estimate the level of user interests in the future accesses. Such major products in the market like Miaopai and Kuaishou make personalized content recommendations according to the user interest. Although existing studies have investigated how users express interests by examining and understanding various user behaviors [33]–[35], we may push it one step further–predicting the future user behaviors based on the potential user interests, where machine learning techniques can be applied.

VII. Conclusion

In this paper, we presented an initial study on instant video clip sharing services enabled by mobile platforms and explored the potentials for its further enhancement. Taking Vine as an example, we closely investigated its mobile user interface and system architecture, and examined underlying services that enable this mobile social application. We presented the key difference between mobile instant video sharing services and traditional video sharing services, including highly skewed popularity, fast propagation, short lifespan. We further identified and characterized the unique watching behaviors of this mainstream multimedia service type, namely, batch view, passive view and screen scrolling. To enhance Vine-like services, we formulated a generic scheduling problem to maximize the viewing experience as well as the cost efficiency, which is shown to be NP-complete. To better solve it, we further divided the problem into two subproblems, specifically, the pre-fetching scheduling and the watch-time download scheduling, conquered them separately and then developed a general solution for the generic problem. Using extensive simulations driven by the real-world traces, we showed that our solution can significantly improve the viewing experience while still keeping high cost efficiency.

Appendix A
Proof of Theorem 1

The corresponding decision problem can be described as: given all the required parameters, is there a schedule for the playlist such that the objective value given by (1) is at most M? First, we show that this decision problem is in NP. Given an instance of this decision problem, a certificate that it is solvable would be a specification of the downloading schedules for each video. We can then easily check each video's playback discontinuity, downloading monetary cost, energy consumption and whether the objective value is no greater than M, and thus verify the solution in polynomial time, which suggests the decision problem is in NP.

We next show that the Knapsack problem is reducible to our problem. The decision version of the Knapsack problem can be stated as: given n items with size $\hat{l}_1, ..., \hat{l}_n$ and value $\hat{s}_1, ..., \hat{s}_n$, capacity W and value S, is there a subset $I \subseteq 1, 2, ..., n$ such that $\sum_{i \in I} \hat{l}_i \leq W$ and $\sum_{i \in I} \hat{s}_i \geq S$? To construct an equivalent scheduling instance of our problem, one may be struck initially by the fact that we have so many parameters to manage. The key is to sacrifice some of the flexibility, producing a simpler "skeletal" instance of the problem that still encodes the Knapsack problem. Let $p = 1$, $q = 0$ and $r = 0$ in (1). The objective of optimization problem is thus to minimize Discontinuity of the playlist V, which is equivalent to maximize $1 - \text{Discontinuity} = 1 - \sum_{v_i \in V} w_i \cdot \text{discontinuity}(v_i) = \sum_{v_i \in V} w_i \cdot (1 - \text{discontinuity}(v_i))$, given that $\text{Discontinuity} \in [0, 1]$, $\text{discontinuity}(v_i) \in [0, 1]$. Let $B_{3/4G} = B_{\text{wifi}} = B$, which implies that we disregard the difference of connection type. Assume that all the available downloading slots exist before the watch-time, which suggests that, instead of producing detailed download schedules, we only need to make download decisions (i.e., different downloading times make no difference).

Given the Knapsack instance, we now show how to convert it to an instance of our problem in polynomial time.

Corresponding to the capacity W and the n items in the Knapsack problem, we have W downloading slots and n videos $v_1, ..., v_n \in V$. Assume the watch duration (effective length) for each video v_i is 1 time slot. The problem then becomes to decide whether to download each v_i (i.e., discontinuity$(v_i) \in \{0, 1\}$). We set the streaming rate of v_i as $R_i = \hat{l}_i \cdot B$, so that downloading v_i takes $\hat{l}_i$ time slots. Note that, in our problem, the playlist discontinuity is a weighted sum of individual video discontinuity. We set the weight of v_i as $w_i = \hat{s}_i / \sum_{j\in[1,n]} \hat{s}_j$, and $M = 1 - S/\sum_{j\in[1,n]} \hat{s}_j$. Now our problem is to download the videos with the given W available time slots such that $\sum_{i\in[1,n]} w_i \cdot x_i \geq 1 - M = S/\sum_{j\in[1,n]} \hat{s}_j$, where $x_i = 1 -$ discontinuity(v_i) is 1 if v_i is downloaded and 0 otherwise. This described instance is equivalent to the original Knapsack decision problem except the value for each item is scaled down by a constant of $\sum_{j\in[1,n]} \hat{s}_j$.

Consider any instance that satisfies (answers "Yes" to) the Knapsack decision problem with the chosen subset I. In our scheduling problem, we download videos with indices in I, which suggests discontinuity$(v_i) = 0, \forall i \in I$ and $x_i = 1 -$ discontinuity$(v_i) = 1, \forall i \in I$. This download schedule (downloading videos with indices in I) uses at most W time slots since $\sum_{i\in I} \hat{l}_i \leq W$. The objective value given by (1) is Discontinuity $= \sum_{i\in[1,n]} w_i \cdot$ discontinuity$(v_i) = \sum_{i\in[1,n]} w_i \cdot (1 - x_i) = 1 - \sum_{i\in[1,n]} w_i \cdot x_i = 1 - \sum_{i\in[1,n]} x_i \cdot \hat{s}_i / \sum_{j\in[1,n]} \hat{s}_j = 1 - \sum_{i\in I} \hat{s}_i / \sum_{j\in[1,n]} \hat{s}_j \leq 1 - S/\sum_{j\in[1,n]} \hat{s}_j = M$. Therefore, downloading videos with indices in I satisfies (answers "Yes" to) our decision problem.

Conversely, if there is a schedule (a set of download decisions) for our constructed decision problem instance such that Discontinuity $\leq M$. The subset I for the Knapsack decision problem can be defined as the set of indices of the videos that are downloaded ($i \in I, if\ v_i$ is downloaded). Since $S/\sum_{j\in[1,n]} \hat{s}_j = 1 - M \leq 1 -$ Discontinuity $= 1 - \sum_{i\in[1,n]} w_i \cdot$ discontinuity$(v_i) = \sum_{i\in[1,n]} w_i \cdot (1 -$ discontinuity$(v_i)) = \sum_{i\in I} w_i = \sum_{i\in I} \hat{s}_i / \sum_{j\in[1,n]} \hat{s}_j$, we can have $\sum_{i\in I} \hat{s}_i \geq S$. As the schedule is valid, which uses at most W downloading slots, we naturally have $\sum_{i\in I} \hat{l}_i \leq W$. Therefore, this subset I satisfies (answers "Yes" to) the Knapsack decision problem. This finishes the proof that the decision version of our original modeled optimization problem is NP-complete.

ACKNOWLEDGMENT

The findings achieved herein are solely the responsibility of the authors.

REFERENCES

[1] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. ACM Internet Meas. Conf.*, 2007, pp. 1–14.

[2] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of internet short video sharing: A youtube-based measurement study," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1184–1194, Aug. 2013.

[3] T. Rodrigues, F. Benevenuto, M. Cha, K. Gummadi, and V. Almeida, "On word-of-mouth based discovery of the web," in *Proc. ACM Internet Meas. Conf.*, 2011, pp. 381–396.

[4] H. Li, H. Wang, J. Liu, and K. Xu, "Video sharing in online social networks: measurement and analysis," in *Proc. ACM Netw. Oper. Syst. Support Dig. Audio Video*, 2012, pp. 83–88.

[5] S. Yarosh, E. Bonsignore, S. McRoberts, and T. Peyton, "Youthtube: Youth video authorship on youtube and vine," in *Proc. ACM Conf. Comput. Support. Cooperative Work*, 2016, pp. 1423–1437.

[6] N. Gautam, H. Petander, and J. Noel, "A comparison of the cost and energy efficiency of prefetching and streaming of mobile video," in *Proc. ACM Workshop Mobile Video*, 2013, pp. 7–12.

[7] L. Zhang, F. Wang, J. Liu, and X. Ma, "On mobile instant video clip sharing with screen scrolling," in *Proc. IEEE/ACM Int. Symp. Quality Serv.*, 2016, pp. 1–10.

[8] Z. Wang *et al.*, "Propagation-and mobility-aware D2D social content replication," *IEEE Trans. Mobile Comput.*, vol. 16, no. 4, pp. 1107–1120, Apr. 2016.

[9] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. ACM Internet Meas. Conf.*, 2009, pp. 280–293.

[10] J. Huang *et al.*, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM Int. Conf. Mobile Syst., Appl., Serv.*, 2012, pp. 225–238.

[11] X. Cheng, H. Li, and J. Liu, "Video sharing propagation in social networks: Measurement, modeling, and analysis," in *Proc. IEEE INFOCOM*, 2013, pp. 45–49.

[12] Y. Xu and F. Liu, "Qos provisionings for device-to-device content delivery in cellular networks," *IEEE Trans. Multimedia*, vol. 19, no. 11, pp. 2597–2608, Nov. 2017.

[13] D. He *et al.*, "Progressive pseudo-analog transmission for mobile video streaming," *IEEE Trans. Multimedia*, vol. 19, no. 8, pp. 1894–1907, Aug. 2017.

[14] R. Trestian, A.-N. Moldovan, O. Ormond, and G. Muntean, "Energy consumption analysis of video streaming to android mobile devices," in *Proc. IEEE Netw. Oper. Manage. Symp.*, 2012, pp. 444–452.

[15] T. Schierl, T. Stockhammer, and T. Wiegand, "Mobile video transmission using scalable video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1204–1217, Sep. 2007.

[16] Z. Yuan, G. Ghinea, and G.-M. Muntean, "Beyond multimedia adaptation: Quality of experience-aware multi-sensorial media delivery," *IEEE Trans. Multimedia*, vol. 17, no. 1, pp. 104–117, Jan. 2015.

[17] Y. Wen, X. Zhu, J. J. Rodrigues, and C. W. Chen, "Cloud mobile media: Reflections and outlook," *IEEE Trans. Multimedia*, vol. 16, no. 4, pp. 885–902, Jun. 2014.

[18] X. Wang, M. Chen, T. T. Kwon, L. Yang, and V. Leung, "AMES-cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 811–820, Jun. 2013.

[19] Z. Wang, B. Li, L. Sun, W. Zhu, and S. Yang, "Dispersing instant social video service across multiple clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 3, pp. 735–747, Mar. 2016.

[20] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over http in vehicular environments," in *Proc. ACM Workshop Mobile Video*, 2012, pp. 37–42.

[21] D. De Vleeschauwer *et al.*, "Optimization of http adaptive streaming over mobile cellular networks," in *Proc. IEEE INFOCOM*, 2013, pp. 898–997.

[22] H. Lu *et al.*, "The jigsaw continuous sensing engine for mobile phone applications," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2010, pp. 71–84.

[23] S. Nath, "Ace: Exploiting correlation for energy-efficient and continuous context sensing," in *Proc. ACM Int. Conf. Mobile Syst., Appl., Serv.*, 2012, pp. 29–42.

[24] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. ACM KDD*, 2011, pp. 1082–1090.

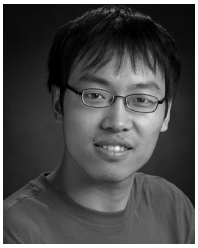
[25] K. Evensen *et al.*, "Mobile video streaming using location-based network prediction and transparent handover," in *Proc. ACM Netw. Oper. Syst. Support Dig. Audio Video*, 2011, pp. 21–26.

[26] R. K. Panta, "Mobile video delivery: Challenges and opportunities," *IEEE Internet Comput.*, vol. 19, no. 3, pp. 64–67, May/Jun. 2015.

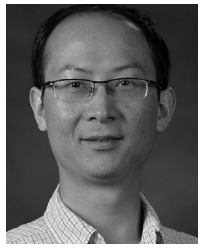
[27] A. Roy, P. De, and N. Saxena, "Location-based social video sharing over next generation cellular networks," *IEEE Commun. Mag.*, vol. 53, no. 10, pp. 136–143, Oct. 2015.

[28] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Commun. ACM*, vol. 53, no. 8, pp. 80–88, 2010.

[29] Z. Wang, L. Sun, C. Wu, and S. Yang, "Guiding internet-scale video service deployment using microblog-based prediction," in *Proc. IEEE INFOCOM*, 2012, pp. 2901–2905.
[30] D. A. Shamma, J. Yew, L. Kennedy, and E. F. Churchill, "Viral actions: Predicting video view counts using synchronous sharing behaviors," in *Int. AAAI Conf. Weblogs Social Media*, pp. 618–621, 2011.
[31] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2011, pp. 177–186.
[32] J. Xu, M. Van Der Schaar, J. Liu, and H. Li, "Timely video popularity forecasting based on social networks," in *Proc. IEEE INFOCOM*, 2015, pp. 2308–2316.
[33] R. W. White, P. Bailey, and L. Chen, "Predicting user interests from contextual information," in *Proc. ACM SIGIR*, 2009, pp. 363–370.
[34] J. Ruiz, Y. Li, and E. Lank, "User-defined motion gestures for mobile interaction," in *Proc. ACM CHI*, 2011, pp. 197–206.
[35] L. Sun, X. Wang, Z. Wang, H. V. Zhao, and W. Zhu, "Social-aware video recommendation for online social groups," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 609–618, Mar. 2017.



Lei Zhang (S'12) received the B.Eng. degree in 2011 from the Advanced Class of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, and the M.S. degree in 2013 from Simon Fraser University, Burnaby, BC, Canada, where he is currently working toward the Ph.D. degree at the School of Computing Science. His research interests include mobile cloud computing, social media, and multimedia systems and networks. Mr. Zhang is a recipient of the C.D. Nelson Memorial Graduate Scholarship (2013).



Feng Wang (S'07–M'13) received both the Bachelor's and Master's degrees in computer science and technology from Tsinghua University, Beijing, China, in 2002 and 2005, respectively, and the Ph.D. degree in computing science from Simon Fraser University, Burnaby, BC, Canada, in 2012. He is currently an Assistant Professor with the Department of Computer and Information Science, University of Mississippi, University, MS, USA. His research interests include wireless mesh/sensor networks, cyber-physical systems, peer-to-peer networks, socialized content sharing, cloud computing, and big data. Prof. Wang is the recipient of the Chinese Government Scholarship for Outstanding Self-financed Students Studying Abroad in 2009 and the IEEE ICME Quality Reviewer Award in 2011. He is a Technical Committee Member of Elsevier Computer Communications. He served as Program Vice Chair in the International Conference on Internet of Vehicles 2014, and as TPC Co-Chair in the IEEE CloudCom 2017 for Internet of things and mobile on cloud track. He also serves as TPC member in various international conferences such as the IEEE INFOCOM, ICPP, the IEEE/ACM IWQoS, ACM Multimedia, the IEEE ICC, the IEEE GLOBECOM, and the IEEE ICME.

Jiangchuan Liu (S'01–M'03–SM'08–F'17) received the B.Eng. degree (*cum laude*) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2003, both in computer science. He is a University Professor in the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. He is an EMC-Endowed Visiting Chair Professor of Tsinghua University, Beijing, China, and an Adjunct Professor of Tsinghua-Berkeley Shenzhen Institute, Berkeley, CA, USA. His research interests include multimedia systems and networks, cloud computing, social networking, online gaming, big data computing, RFID, and Internet of things. Prof. Liu is an NSERC E.W.R. Steacie Memorial Fellow. He is a co-recipient of the inaugural Test of Time Paper Award of the IEEE INFOCOM in 2015, ACM SIGMM TOMCCAP Nicolas D. Georganas Best Paper Award in 2013, and ACM Multimedia Best Paper Award in 2012. He has served on the editorial boards of the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON BIG DATA, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and the *IEEE Internet of Things Journal*. He is a Steering Committee member of IEEE TRANSACTIONS ON MOBILE COMPUTING and Steering Committee Chair of IEEE/ACM IWQoS (2015–2017).