Christian Koch, Benedikt Lins, Amr Rizk, Ralf Steinmetz and David Hausheer.
vFetch: Video Prefetching using Pseudo Subscriptions and User Channel Affinity in YouTube
In Proc. of the IFIP/IEEE International Conference on Network and Service Management (CNSM), 2017

# vFetch: Video Prefetching using Pseudo Subscriptions and User Channel Affinity in YouTube

Christian Koch*, Benedikt Lins*, Amr Rizk*, Ralf Steinmetz* and David Hausheer‡
* Multimedia Communications Lab, Technische Universität Darmstadt, Germany
Email: {Christian.Koch|Benedikt.Lins|Amr.Rizk|Ralf.Steinmetz}@kom.tu-darmstadt.de
‡ Otto-von-Guericke University Magdeburg, Germany, Email: hausheer@ovgu.de

***Abstract*—Video streaming is responsible for the largest portion of traffic in fixed and mobile networks. Yet, forecasts expect this amount to grow further. Especially for mobile devices connected to cellular networks, high QoE video streaming can be a challenge as the user data volume is metered and eventually limited. Also, the connection quality may vary severely. Prefetching videos is an approach to mitigate this issue. Here, videos that the user is likely to watch in advance are prefetched on the user's smartphone, e.g., while he is connected to WiFi. However, this approach can only be efficient if *only* the videos that are interesting for the respective user are prefetched. This constitutes a major estimation and prediction challenge. To this end, this paper presents three contributions: First, a user study over multiple months that draws valuable insights on the user video request behavior. Second, we propose a novel privacy-preserving prefetching framework denoted vFetch that prefetches videos based, e.g., on the user's affinity of YouTube channels. Third, a trace-based evaluation and parameter study that demonstrates vFetch's efficiency with a hit rate of ∼50% for a 50 GB cache.**

## I. Introduction

Video streaming causes the majority of Internet traffic nowadays. Popular services like YouTube, Netflix, and Amazon Prime accelerate this trend. While the latter ones are used to stream professionally produced movies or series, YouTube offers more content that comprises especially short videos. Also, YouTube represents the largest single source of user-generated traffic estimated to range between 40% and 70% in most networks [1]. Together with mobile data rates, also video traffic grows, caused by an increasing number of mobile devices, usage of video streaming services, and higher video qualities. However, the user perceived Quality of Experience (QoE) strongly depends on the available bandwidth. A reduced throughput leads to shorter video playback duration [2] and even playback abortion [3] if the playback does not start promptly. Allowing a fast video playback start without interruptions in the presence of steeply increasing traffic volumes states a major challenge on network management. Addressing this challenge, e.g., WiFi and femto cell offloading are proposed and state recent research topics envisioned as key elements in efficient content dissemination and network management solutions. Preliminary work has shown that prefetching on servers [4], [5], femto cells [6], home routers [7], [8], and user terminals [9], [10] has the potential to efficiently unburden networks by offloading video traffic. While it is quite likely that a user watches the next episode of a series if he has watched the preceding ones, this is not as easy for services like YouTube. The intuitive idea of prefetching recent videos from subscribed YouTube channels is not sufficient, as we will show in the data analysis section of this paper. Also selecting videos for prefetching recommended by YouTube [11] and the video's like count [12] have shown a poor performance. Hence, we conclude that personalized prefetching of videos is still a challenging task since native features, such as the subscription status and the global popularity, are usually rather ineffective for prefetching algorithms. To this end, we conducted a months-long user study and analyzed user behavior to deduce requirements on an efficient prefetching system. In the next step, this paper proposes **vFetch**, a **v**ideo pre**fetch**ing system for efficient video selection based on content features and user channel affinity. This allows efficient content placement of delay-tolerant videos in the context of prefetching. Hence, traffic peaks can be reduced and the traffic pattern be smoothened, which reduces transit costs that arise due to burstable billing. To this end, this paper presents three contributions: First, we conducted a user study using a dedicated Android app and derived insights from a thorough analysis giving guidance on how to design a prefetching system. Second, using these insights, we design a novel prefetching system named vFetch considering pseudo subscriptions and user channel affinity. Third, using the user study participants' video requests, we conduct trace-based simulations to demonstrate the efficiency of vFetch and discuss key parameter configurations.

The remainder of this paper is structured as follows. Sec. II gives an overview of relevant background and related work. In Sec. III, the user study conducted is described. An analysis of the user study is conducted in Sec. IV. Sec. V describes the design of vFetch, which is evaluated in Sec. VI. Sec. VII concludes the paper and discusses future work.

## II. Background and Related Work

The two major goals of video prefetching are *(i)* network offloading by downloading videos at off-peak hours to alleviate networks at peak times and *(ii)* to avoid bad or fluctuating QoE due to unstable cellular connection by downloading videos in advance. Consequently, the QoE perceived by the user can be increased as bandwidth and coverage often state problems for mobile users watching videos, e.g., while they are commuting by train. In the following, we discuss the key areas for the application of prefetching and most relevant works.

978-3-901882-98-2 

Kaafar et al. [13] propose a prefetching approach for CDNs. Thereby, prefetching candidates are determined based on a recommendation approach. Summers et al. [4] analyze Netflix's workloads showing chains of sequential requests. Here, the authors present prefetching algorithms that reduce the hard disk and main memory utilization, thereby not focusing on individual users. Golrezaei et al. [6] investigate the potential of prefetching at femto cells for cooperative caching in mobile networks to relieve the back-haul network. Bai et al. [14] discuss caching mechanisms using information from Online Social Networks (OSNs) for content published on Facebook and Yahoo. These approaches propose solutions for social-aware prefetching in the context of caching within the network, e.g., a CDN or an ISP cache while addressing many users. In contrast to this, vFetch addresses user-specific prefetching, i.e., individually downloading video content directly to the user's storage, e.g., on a home router or the user's smartphone.

Two prominent works discuss the application of prefetching to peer-to-peer (P2P) systems. Wang et al. [15] use RenRen, the Chinese Facebook variant to investigate the benefits achievable through P2P-assisted video streaming. Their goal is to reduce the initial video buffering time to increase the user QoE. Therefore, the authors proposed prefetching the first segment of videos, which are likely to be watched. To determine these videos, the authors use social relationships and user preferences as predictive features. Li et al. present SocialTube [16], a P2P video prefetching system focusing on social relationships. The authors observed that 0.4% of the videos cause 80% of the traffic. Consequently, 99.6% of the videos cause only 20% of the network traffic. A further interesting observation is that 90% of a user's video views can be explained by direct and 2-hops friends. All studied users watch at least 20% of the videos on their social network feeds, while 33% even watch 80% of these videos.

While the previously presented approaches do not address prefetching on mobile devices, Zhao et al. [17] develop a customized Facebook application. Thereby, they investigate social network-based prefetching algorithms. However, the findings here are limited and require validation through simulation experiments. Closing this gap, Gouta et al. [10] conduct simulations for prefetching on mobile devices using a trace from a mobile network operator. The authors use a collaborative filtering approach inspired by Google's page rank algorithm to predict videos for users based on user similarity. However, the approach requires continuous monitoring of many users to work efficiently. This violates the user privacy. In contrast to this, vFetch is designed to work on a user device only using the respective user's data, which remain local not leaving his device. In a previous work [5], we investigated recommender approaches for music video prefetching on network caches and mobile devices. Therefore, we used a dataset covering two weeks for several thousand users, which does not allow analyzing long-term user behavior and requires a continuous user request monitoring. Plecsca et al. [9] leverage the users' tendency to request videos related to the last video they watched, e.g., by selecting videos from the related video list of YouTube. To leverage this observation, the authors proposed prefetching based on Markovian policies fed by the YouTube video recommendation list. While this behavior is valid, especially for fixed networks without mobile data caps, the challenge of prefetching the first video in a video session not addressed. Wilk et al. [12] present an analysis of a user study focusing on Facebook. In their user study with 14 participants, they investigate social-aware multimedia prefetching by providing an Android app called SonNet to the participants. The results show that video consumption is quite diverse across users and that there is no single predictor applicable for all users. Especially the number of likes, comments, and the content coming from a 1-hop-friend alone are not sufficiently predictive features. Therefore, this work focuses on video content features instead of social relationships. Furthermore, Facebook API changes do not allow using the SonNet app anymore. Summarizing, Zhao et al. [17] present the most promising approach, where a customized Facebook-like app has been developed using Facebook-based prefetching algorithms. Unfortunately, it remains unclear how the app orders Facebook posts and, generally, which design choices the authors made to create the framework.

In contrast to the existing prefetching approaches, this work differs in three aspects. First, we do not require the user to install a customized app that changes look and feel for the user but rely on a seamless monitoring app, allowing usage of the native YouTube app. Second, the user study conducted uses the YouTube watch history from 27 active YouTube users allowing us to study their individual behavior over several months. Third, our aim to investigating prefetching policies for individual users does not require extensive user traces as necessary for non-privacy-friendly recommender approaches. Instead, we design a simulator that is able to evaluate different prefetching policies based on the collected ground-truth user requests. Thereby, we consider the substantial contribution on the case of key design aspects of user-specific prefetching mechanisms. This is useful to dimension a plenitude of use cases, e.g., YouTube apps, smartphone middlewares, or the operation of cellular cloudlets.

## III. User Study

In order to collect real world user data, we developed an Android app, denoted as *SocialMonitor*. We handed this app out to YouTube users within the scope of a user study, that startet in January 2015. We managed to acquire additional participants in the following semesters, by asking students in lectures as well as in the scope of theses to participate. After the first start, the app informs the user about which data is collected and for which purpose. If the user does not agree with the privacy consent, no data is collected. The *SocialMonitor* anonymizes all personal data on the user's device using a cryptographic hash function, before uploading it to a trace collection server. The app users' origin is mostly Germany, India, and France and 58% are male. Their average age is 24.5 with a standard deviation of 10.2 years. The architecture of the *SocialMonitor* is depicted by Fig. 1.

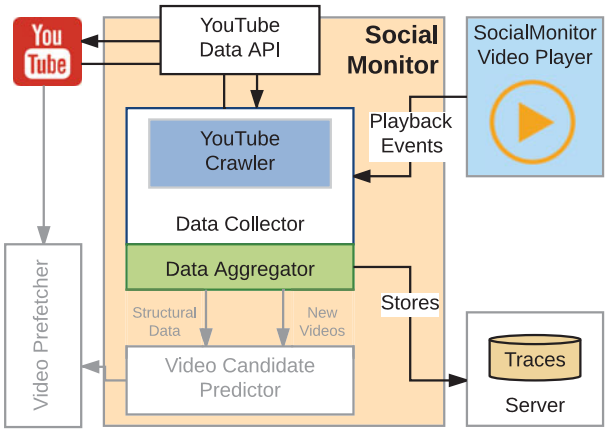


Fig. 1: Architecture of the *SocialMonitor* App

Here, the *Data Collector* collects information on channel un-/subscribe events, as well as the videos watched by the user according to their YouTube watch history. This information is passed to the *Data Aggregator*, which filters, anonymizes, and locally stores the data. Once per hour, if the device has a WiFi connection, *SocialMonitor* sends the data to the traces server, where the data of all participants is stored. If no WiFi is available, the data is stored at most three days before it is sent via cellular connection. Thereby, all videos watched while the user is logged-in using, e.g., the smartphone, browser, or smart TV are collected. It is important to note that by using YouTube's watch history it cannot be determined on which device a user watched a video. However, this information is not necessary to determine user-specific prefetching policies in general. The collected video IDs allow to derive more features, e.g., the title, numbers of likes, the upload date, and the name of the uploading channel. Next, the channel name is used to determine all videos published while a user has subscribed to the channel by using the YouTube Data API.

In the following, we only consider participants that watched at least two videos per day on average and participated at least two weeks in the user study. Furthermore, we only consider devices with less than 10% of the watched videos being deleted later on by YouTube, e.g., because of copyright infringement. This results in 27 user devices, in the following denoted as participants or users. Thereby, the number of our participants and duration surpasses the related works with 10-15 users observed over 10 days - 8 weeks [11], [12], [17]. Statistics on our participants are shown in Table I showing that users are quite diverse w.r.t. the number of days they participated in the study, the number of the subscribed channels (#Subscriptions), and the number of channels they requested videos from (#Channels watched). On average, our users participated 145.8 days in the study and watched videos on 123 days. The videos came from 464.5 channels (median value) with a large std. of about 763.9 channels. The mean number of subscribed channels is 23.9 and 11.8% of the videos coming from subscriptions were watched by the participants, while the max. value is 70.8%. Complementing Fig. 2, the row *%Watched Subs* shows detailed statistics of the share of watched videos from subscribed channels.

## IV. Dataset Analysis

In the following, the user behavior relevant for prefetching of the selected 27 users is further analyzed, see Table II. All of the following statements refer to a daily scale and give a general overview of the participants' behavior. On average, 9.7 videos are provided to the users by subscriptions. Overall, the participants watch 10.5 videos from 7.2 channels. However, the participants watch only 16.7 of the videos offered from subscriptions on average and 83.3 from other sources. Please note that we calculated the *Watch time* by summing the duration of the videos watched, resulting in a median value of 74.6 minutes per day. However, the maximum value exceeds the minutes of the day indicating that this user did not watch the videos completely but skipped videos, which is a common behavior for YouTube users observed before [18].

### A. Subscriptions

Fig. 2 depicts the CDF of the share of watched videos from subscribed channels. Note that the participants of our user study did not watch most videos from their subscriptions. Surprisingly, only for about 7% of the subscribed channels, the participants are interested enough to watch all videos. Further, for 19% of subscribed channels, no video is watched and 80% of the subscribed channels are watched 34% or less. Hence, we conclude that the channel subscription status alone is not an effective feature for video prefetching. Consequently, the ratio of videos watched from a subscribed channel has to be monitored to determine if videos from it should be prefetched.

TABLE I: Statistics of user study participants

| | Mean | Median | StdDev | Min | Max |
|---|---|---|---|---|---|
| **Participation (days)** | 219.5 | 145.8 | 193.4 | 26.2 | 639.8 |
| **#Days with views** | 151.7 | 123 | 129.2 | 22 | 449 |
| **#Subscriptions** | 23.9 | 10 | 43.2 | 0 | 172 |
| **#Channels watched** | 882.6 | 464.5 | 763.9 | 8 | 2,410 |
| **%Watched Subs.** | 11.8% | 5.3% | 17.1% | 0% | 70.8% |
| **#Subscribe events** | 24 | 11 | 43.4 | 0 | 173 |
| **#Unsubscribe events** | 23.9 | 10 | 43.2 | 0 | 172 |

TABLE II: Daily user statistics

| | Mean | Median | StdDev | Min | Max |
|---|---|---|---|---|---|
| **Subs. videos offered** | 9.7 | 5 | 69.0 | 0 | 6,293 |
| **Number of views** | 10.5 | 8.1 | 12.2 | 2.9 | 69 |
| **#Subscription views** | 3.6 | 0.7 | 10.1 | 0.1 | 48.32 |
| **#Channels watched** | 7.2 | 5 | 8.5 | 0 | 80 |
| **%Subs. watched** | 16.7% | 0% | 28.0% | 0% | 100% |
| **%Others watched** | 83.3% | 100% | 28.0% | 0% | 100% |
| **Watch time (min.)** | 172.7 | 74.6 | 269.3 | 0.15 | 3,644 |

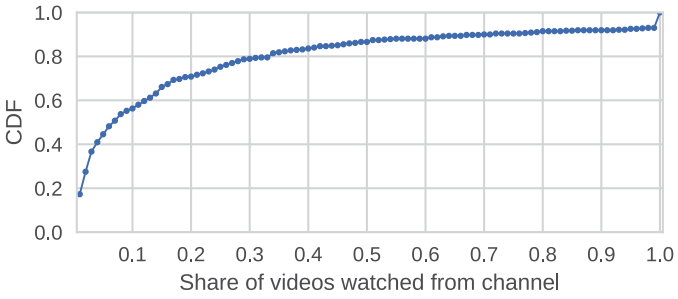


Fig. 2: CDF of videos watched from subscribed channels

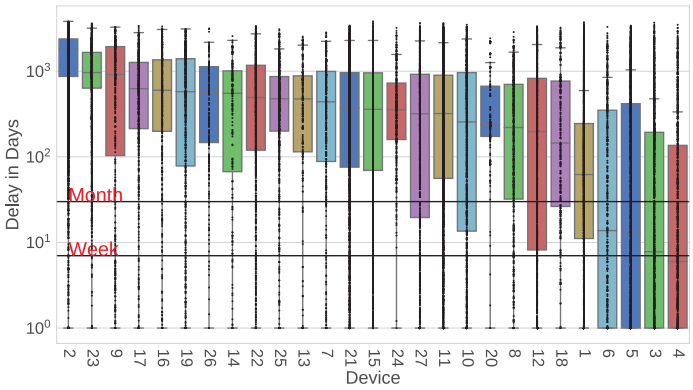


Fig. 3: Age of all videos watched for distinct users (days)

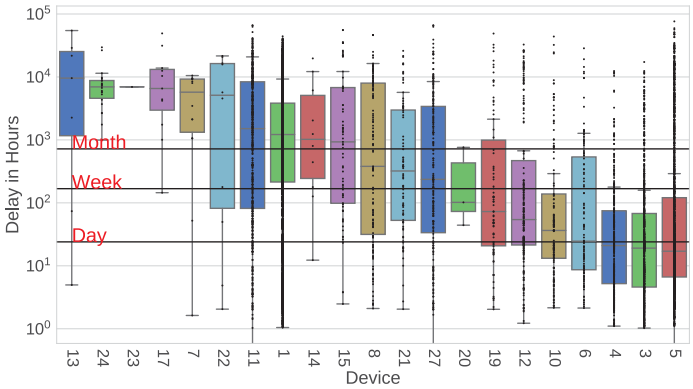


Fig. 4: Age of subscription videos watched per users (hours)

### B. Video Age

To constrain the set of videos which come into consideration for prefetching, we analyzed the video age, i.e., the time between the video was uploaded and the time it was watched by the participants. Fig. 3 depicts the video age distribution in days per participant. In all of the following figures, the user ID ranging between 1–27 is assigned by the number of average daily views of the user, i.e., user 1 is most active. While there is a tendency to watch videos within the first two weeks after their upload, for most of the participants, the median shows a time difference between $10^2$ and $10^3$ days. Surprisingly, this indicates that mostly videos older than one month are watched. YouTube video requests can have many reasons, e.g., a new video on a subscribed channel, a search for an ephemeral interest, or a share on Facebook. Therefore, we further looked into the age of videos requested from subscribed channels. Fig. 4 shows that here, the video age is lower compared to the general age of all videos watched. Note that the y-axis measures in hours instead of days. Six participants, present in Fig. 3 are missing in this figure as no videos from subscribed channels were watched. About half of the participants watch subscribed videos with a median age of around $10^3$ hours, i.e., 42 days. For 7 participants the median was even smaller than $10^2$ hours, i.e., 4 days, showing a diverse behavior among the participants. However, for distinct participants the second and third quartile are narrow, which suggests a stable per-user behavior. Dividing the participants' standard deviation by their mean results in values smaller than 9 for all participants except participants 3, 4, 5, and 6 which range between 38 and 80. Hence, the user behavior is quite constant. Hence, we conclude that videos from subscriptions have to be prefetched within the first 7 hours after their upload to not miss more than 25% of videos requested, e.g., by participants 3, 4, 5, and 6.

### C. Video Origin

As mentioned previously, a video request can have multiple origins. From the data at hand, we can only determine if a video belongs to a subscribed channel or not. However, we observed substantial video views from participants to channels they have not subscribed but have a non-ephemeral interest in. Hereafter, we refer to these channels as *pseudo subscriptions*. Hence, we distinguish between three video categories: *1)* The video is published on a channel that a user has subscribed to. *2)* The video comes from a channel that the user regularly watches videos from but did not subscribe to, e.g., recommended by the YouTube landing page. *3)* The video does not belong to the previous two categories and, therefore is considered as a random view and may come from a website embedding, a YouTube search, or a recommendation on YouTube landing page, which however matches an ephemeral user interest. To distinguish between categories 2 and 3, we set the following two criteria for a match with category 2: First, the user has watched at least two videos of this channel. Second, the user has at least watched 80% of the videos published on this channel. Hence, the first video watched does not render the channel to a pseudo subscription. Note that the channel status may change from category 2 to 3 and the other way around over time as user interests may change. Fig. 5 shows the share of video sources, i.e., the categories from above, for our participants. Their avg. number of daily video views, as depicted on the x-axis, sorts the participants depicted. The green share represents videos from subscriptions. The yellow share belongs to pseudo subscriptions. The remainder of requests do not belong to subscriptions and not to pseudo subscriptions and thereby explain the gap to 1. Surprisingly, subscriptions and pseudo subscriptions cannot explain most requests, which, therefore seem to be random. However, subscriptions are responsible for about 10% of views for most participants with less than

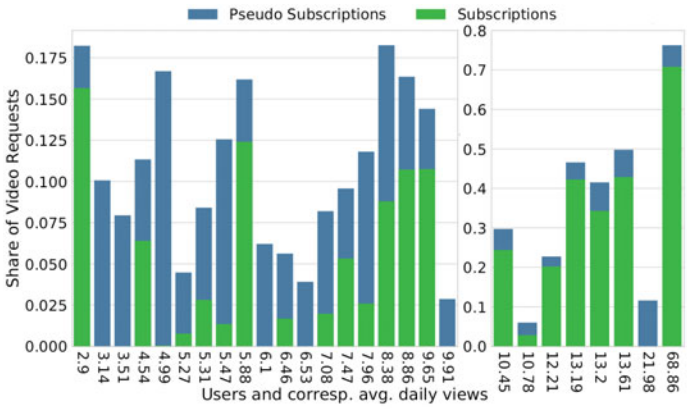


Fig. 5: Share of videos requested per subscribed (green) and pseudo subscribed channels (blue), stacked per user

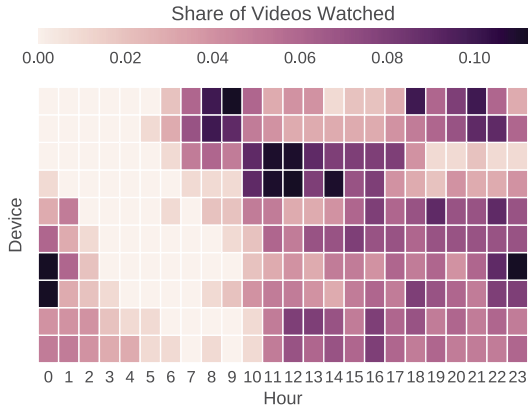


Fig. 7: Views over the hours of the day

8 views a day. For participants with more than 7 views per day, much higher subscription shares are observed, reaching more than 33% for 4 participants which requested more than 10 videos per day on average. Videos can only be prefetched if user interest can be estimated in advance. Hence, we conclude that the bars in Fig. 5 represent the individual maximum prefetching potential, which is quite diverse for different study participants ranging between 3% and 77%.

### D. Repeated Video Requests

For prefetching, videos that are watched multiple times are especially valuable as they increase the cache hit rate (CHR) more than videos that are only requested once. Therefore, we investigated which videos users repeatedly request. Fig. 6 shows the number of videos which have been requested more than once by a user, grouped by their YouTube category. This category can be chosen out of a set of categories by the video uploader and can be retrieved by the YouTube Data API. On the one hand, videos belonging to *Music* and *Entertainment* show a high re-watch behavior. One explanation for this is that these videos belong to entertaining categories. In contrast to this, informative video categories, e.g., *Travel&Events*, *News&Politics*, and *Shows* exhibit rarely repeated requests. *Music* videos show the highest re-watch behavior by a median of 13 times. This confirms the fact that most video requests on YouTube belong to the category *Music* [5] with about 42%. Hence, we conclude that videos belonging to entertaining categories should not be deleted by the prefetching system after being watched. One way to achieve this is by a least-recently used (LRU) managed cache storing prefetched videos and keeps repeatedly requested ones longer in the cache.

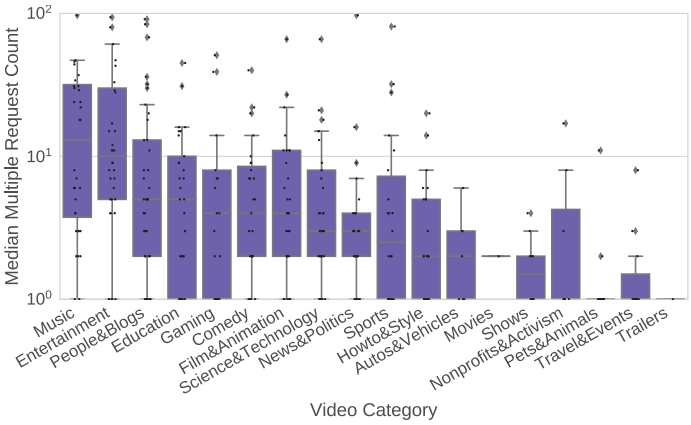


Fig. 6: Videos requested more than once per YouTube category

### E. User Request Time

It is important to know, when a user regularly watches videos to be able to prefetch videos before they are watched. Fig. 7 depicts a heat map of views over the hours of the day, for the ten participants with most average views per day. The fields with the darkest color range from 11% to 17% due to the robust coloring scheme. For all participants, hours exist where they are more likely to watch videos, e.g., 30% of views between 7 and 10 am for the user depicted in the first row. This information is especially important when prefetching for users watching videos briefly after their upload to download them timely and, hence, not miss prefetching opportunities.

## V. vFetch System Design

vFetch's primary goal is allowing user devices such as smartphones, or stationary last mile hardware such as home routers and cloudlets to determine videos to prefetch. Thereby, the user data is processed in a user-owned environment and therefore the user privacy is not violated as it is the case for most recommender systems. To avoid a loss of information on YouTube's side, vFetch sends user viewing statistics to YouTube allowing to distinguish between prefetched and watched videos as well as playback duration and quality information. For the evaluation of this paper, we implemented vFetch to work with the user study participants' traces. Therefore, we use it in combination with a discrete event simulator for distinct user simulations, keeping track of the past and the current simulation time to only use information that is available at the current simulation time, i.e., no information from the future. Fig. 8 presents vFetch's system architecture.

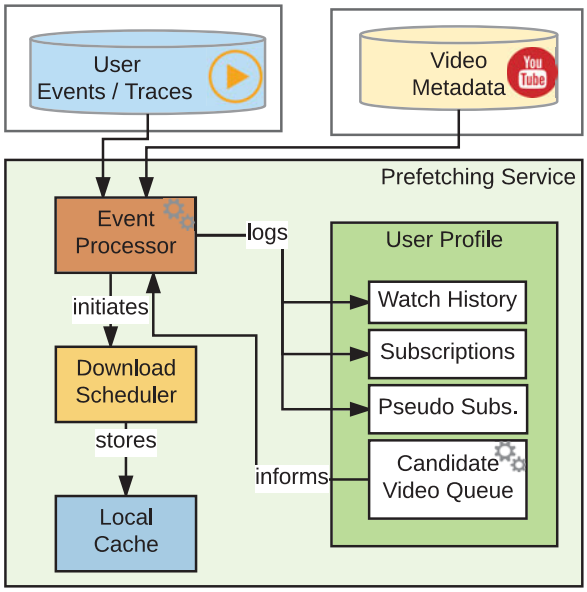


Fig. 8: Architecture of the vFetch Prefetching System

The *User Traces* collected during the user study serve as an input for vFetch. The traces contain information about, e.g., the videos requested by a user as well as un-/subscribe events for YouTube channels. Each video is identified by a video ID, for which further metadata is offered by the *Video Metadata* database, containing, e.g., the title, the description, the upload timestamp as well as the name of the channel on which the video is published. The *Event Processor* processes user events in chronological order, e.g., a new video uploaded by a channel, a video request, or a un-/subscribe event. Furthermore, it retrieves for each video the corresponding metadata from the *Video Metadata* database. The *User Profile* module contains information about the videos watched by the user, i.e., the *Watch History*, *Subscriptions*, and *Pseudo Subscriptions* of the user. Thereby, it captures the user interests. We expect the consideration of pseudo subscriptions as introduced in Sec. IV-C to further increase the efficiency of vFetch. Therefore, only videos published after the start of the users' participation in the user study are considered. To count as a pseudo subscription, a user must watch at least two videos and overall more than 80% of the channel videos uploaded since the user study start. According to [16], 80% is a reasonable value to infer a strong interest for a channel. Based on this information, the *Candidate Video Queue* determines videos that the user is likely to watch by ordering and filtering them appropriately as described in the following. Videos which are older than the threshold $t_{candidate}$ are removed from the candidate queue, as the probability of a corresponding request is low as shown in Sec. IV-B. Within the candidate queue, videos are ordered by the user's affinity to the channel, which has uploaded the video, i.e., the ratio of videos watched from this channel. Therefore, the video candidates with the highest estimated user affinity are prefetched first. In the following, we discuss and reason requirements for vFetch.

### A. Requirements

While a simple FiFo (First In First Out) policy would always put the oldest videos in front of the candidate queue to get them prefetched first, this is not an efficient policy. A substantial part of videos requested is only a few weeks or even hours old, as shown in Sec. IV-B, Therefore, we define requirement *i) Videos older than one month should be removed from the candidate queue, as it is unlikely that the user watches them compared with younger videos.* The next simple behavior we want to discuss is LiFo (Last In First Out), which is likely to have a higher prefetching efficiency. However, just prefetching the most recent videos does not consider the user's heterogeneous channel interests as shown in Fig. 2. Therefore, we define requirement: *ii) The user's affinity to a subscribed channel must be considered by selecting which videos to prefetch.* We implement this by taking the affinity as $\frac{\text{videos watched from channel}}{\text{videos published by channel}}$ for a time span starting with the first request of the user for this channel. Even though the participants subscribed to channels, only 16.7% of videos published by these subscriptions are watched, as show in Table II. For most, but not all participants, the videos watched from not subscribed channels dominate. In a further analysis on video origins presented in Sec. IV-C, we observed that even though the participants did not subscribe to a channel, a significant share, i.e., $\geq$ 80% of videos are watched from distinct channels. Therefore, we define requirement *iii) Pseudo subscriptions must be considered as relevant sources of video views.* For some videos, the participants have a longer lasting interested, leading to repeated requests to Music and Entertainment videos as shown in Sec. IV-D. Therefore, we deduce requirement *iv) Videos should be cached and kept in the cache after they were watched.* This seems to contradict the recommendation of Gouta et al. given in [10] to delete videos after they have been watched from local storage. However, the authors excluded music videos from their analysis which are the most repeatedly watched category of YouTube videos as shown in Sec. IV-D. vFetch fulfills this requirement by filling prefetched videos in an LRU-managed cache so they can be repeatedly locally played back. In order to serve videos by prefetching, they have to be placed timely on the user device. From the analysis results in Sec IV-B and Sec. IV-E, we deduce requirement *v) Videos must be prefetched timely after their upload and before the preferred times of the user watching videos.*

### B. Download Scheduling

The *Event Processor* takes the first entries of the *Candidate Video Queue*, filtered by video age on a regular basis and passes them to the *Download Scheduler*. This module determines when vFetch downloads videos the by keeping track of the user's diurnal connectivity patterns. As the focus of this paper is on general prefetching policy design, we will keep this module simple here and leave application-specific designs for future work. In this work, the *Download Scheduler* uses the clustering algorithm DBSCAN [19] to determine when a user is most likely to watch videos by clustering of

the previous watch times. Thereby, the *Download Scheduler* can download the videos in advance, preferably during off-peak hours or when WiFi is available. An example of ten participants' video watch behavior w.r.t. the hours of the day is given in Sec. IV-E. The time interval of four hours before the estimated user watching time is split into 30-minute segments. If vFetch is running on a smartphone, the *Download Scheduler* can determine the interval with the highest likelihood of an available WiFi connection. Thereby, we consider that prefetching is preferably done via WiFi connections. Once an interval is selected, the *Download Scheduler* assigns up to three prefetching events to this interval. Hence, *vFetch* can prefetch up to 3 videos per 30 minutes. The overall number of videos to prefetch is determined by the avg. number of videos the user watches per day over the preceding, at most, 28 days. Thereby, downloading unnecessarily many videos is avoided. The prefetched videos are stored in a LRU-managed cache. Thereby, vFetch leverages that, e.g., music videos, are likely to be repeatedly requested. vFetch, can be used, e.g., by a smartphone, cloudlet, or a femto cell service, if the user trace information is replaced by real user activities and the video metadata database is replaced by direct calls to the YouTube Data API. Depending on the use case, the download scheduler has to be configured, e.g., depending on the storage resources as well as connectivity and mobility pattern of the user.

## VI. Evaluation

Our evaluation consists of a trace-based simulation using the data gathered within the scope of the presented user study. Due to legal restrictions, we do not allow our app to download videos although this is technically possible as shown in [20]. In the following, we evaluate vFetch's performance based on various parameter configurations.

### A. Storage Size and Caching

In the following, we show the impact of different storage sizes on vFetch's F1-Measure as well as the caching performance given by the Byte Hit Rate (BHR). We choose storage sizes that resemble two cases: (i) vFetch running on modern smartphones, i.e., storage sizes of $\{1, 5, 10, 50\}$ GB, and (ii) vFetch running on cloudlets and storage-equipped home routers that have more storage available, e.g., 100 GB. By choosing larger cache sizes, more prefetched videos can be stored for a longer time.

Fig. 9 depicts the impact of different storage sizes on vFetch's performance. The first row shows the results for videos from predictable sources, i.e., subscriptions, pseudo subscriptions, and watch later list entries. The second row shows the results considering all videos watched, also from unpredictable sources. Furthermore, we distinguish between prefetching and the case when prefetching is combined with request-based caching, i.e., requested videos are placed in an LRU cache additionally to the prefetched ones. In both rows, we show the F1-measure and the Byte Hit Rate (BHR). The F1-measure [21] is the harmonic mean of precision and recall and a robust measure for the quality of the prefetching.

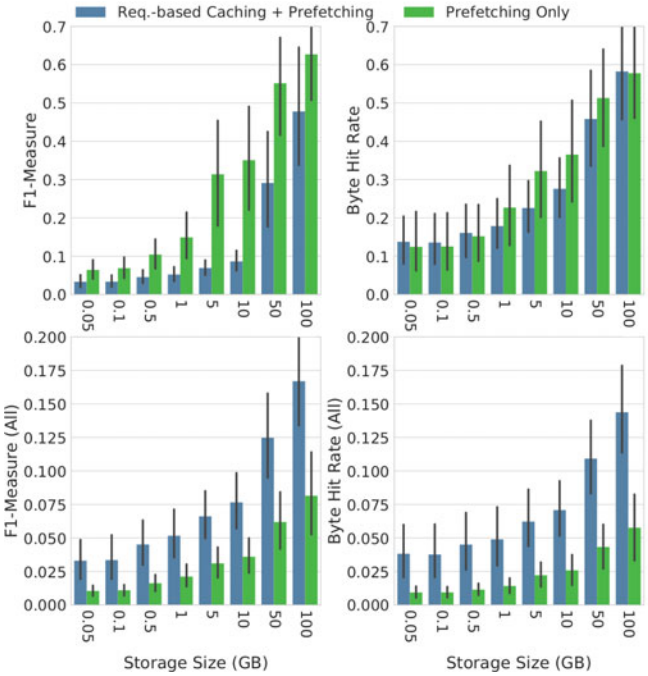


Fig. 9: Performance impact of storage size. top: videos watched from predictable sources, bottom: all videos watched

Precision and recall are about the same for our evaluation, as can be seen by the similarity between F1-measure and BHR, since the BHR is the precision considering repeated requests. From the first row of Fig. 9 it can be observed that given a fixed storage size introducing caching to prefetching severely decreases the F1-measure and the BHR for predictable videos. Considering all requested videos, caching additionally to prefetching increases the performance metrics by a factor of 2.8 on average. This finding shows that predictability contains more valuable information, which is used by prefetching, than the mere object requests, which drive the LRU cache. Overall, for the mixture of predictable and ephemeral user interests, we observe that the combination of prefetching and caching yield the best results. Our insight here is that a differentiation based on predictability of the user interests leads to an adapted use of prefetching and caching.

Since the BHR is the precision of vFetch considering repeated requests, it indicates the number of unnecessary downloaded, i.e., never watched videos. In Fig. 9, the data points comprising the 95% confidence intervals represent the average BHR of each participant. In addition to the depicted results, we also evaluated this configuration while omitting pseudo subscriptions. The results significantly differ, i.e., the median BHR was almost 0 as pseudo subscriptions are the dominant source of videos for many participants. Summarizing, vFetch outperforms existing prefetching mechanisms with a BHR between 0.3% and 14% compared with $\leq$ 0.03% [11], [16], [22] when applied to YouTube as shown in [11].

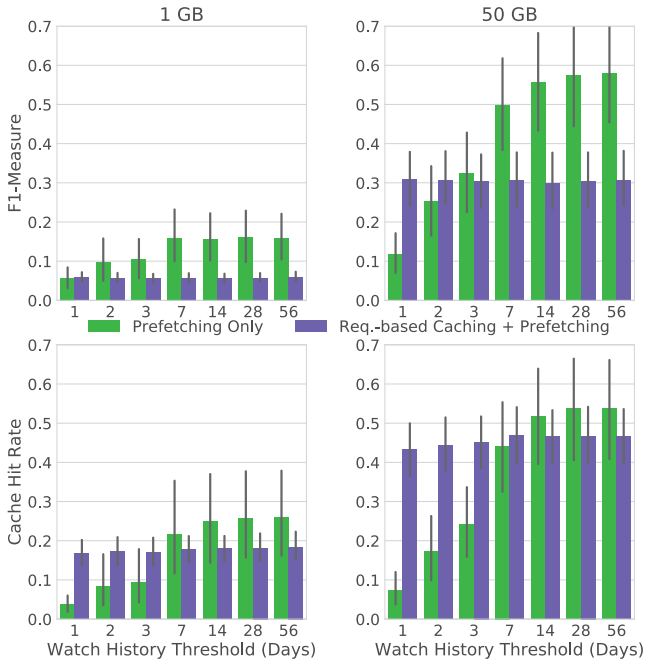


Fig. 10: Performance impact of watch history threshold

### B. Watch History

The watch history is a list of videos the user has watched in the past and can be defined as a time-to-live cache, i.e., the video records are refreshed upon request and remain in this list for a maximum lifetime denoted watch history threshold. By using this threshold, we seek to adapt to the potentially changing request behavior of users, e.g., during holidays or vacation. Furthermore, vFetch determines the number of daily prefetches based on the watch history by using the average number of requested videos per day. Hence, if the user requests fewer videos, less videos are prefetched. Fig. 10 depicts the influence of the watch history threshold on vFetch's F1-measure and Cache Hit Rate (CHR). Here, the first column displays the results for 1GB cache size and the second column the case of 50GB cache size. Additionally, both cases: pure prefetching and request-based caching together with prefetching are depicted. Here, we observe diminishing returns on the F1-measures and the cache hit rates when increasing the watch history threshold. The results indicate that there is no best value for all users; however, we would suggest a window of two weeks to balance performance and length of the watch history. Please note that for window sizes $\geq$ 2 weeks, prefetching surpasses request-based caching in combination with prefetching considering both CHR and F1-measure when benchmarking them against each other.

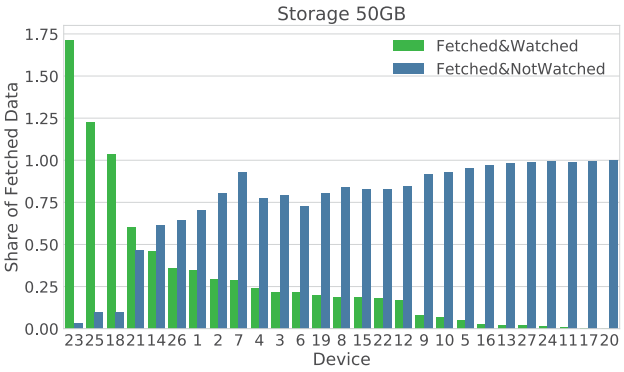


Fig. 11: Share of fetched data watched and not watched

### C. Storage Overhead

In Fig. 11 we depict the share of bytes fetched and watched by a user, as well as, the prefetched and not watched bytes. A value of 100% represents the overall bytes prefetched. Values higher than 100% are possible when users repeatedly watch the same video. It can be seen that the overhead in terms of bytes from prefetched but not watched videos is below 80% for half of the participants. As prefetching is usually performed overnight, e.g., by a cloudlet or by a smartphone connected to WiFi and the charger, we consider an overhead $\leq$ 80% as reasonable for a prefetching system, i.e., $\frac{1}{5}$ of prefetched bytes are consumed. WiFi is about 23 times less power consuming than LTE [23]. Hence, vFetch, with an average overhead of 80%, is about 4-times more energy efficient compared to streaming over LTE. Note that vFetch has a low average overhead of 70%, compared with 82% in case of CPSys [24].

## VII. Conclusion and Future Work

This work investigates the behavior of YouTube users participating in a study over several months. We analyze the user behavior along different axes such as the watched categories, how often, and when the users watch YouTube videos. Further, we show that using pseudo subscriptions as source of user predictability results in a significant performance increase compared to just using videos from subscribed channels and entries of the YouTube *watch later* list. Based on the concluded insights, we derived five requirements for a prefetching system and presented *vFetch*, a novel prefetching system considering user pseudo subscriptions and user channel affinity. Our trace-based evaluation shows the sensitivity of vFetch to key parameters such as the watch history lifetime, different cache sizes, and the performance impact of request caching in addition to prefetching. vFetch achieves an average cache hit rate of 54% on 50 GB storage size. We observe diverse results for different users, indicating that prefetching is efficient for only a subset of our participants, i.e., where predictability can be leveraged. The average overhead of the prefetching system of 70% when applied to all users still remains below the overhead of 82% from the comparable related work in [24]. In future work, we plan a machine learning-based extension of vFetch that is able to identify, e.g., user topic interests and therefore is able to increase the performance of vFetch.

## Acknowledgment

This work has been funded in parts by the DFG as part of the CRC 1053 MAKI (C3, B4) and by the EU FP7/#318398, eCOUSIN. We would like to thank Véronique Henry, Sylvie Vidal, and Yannick Le Louédec for their valuable contribution.

## REFERENCES


[1] Ericsson, "Ericsson Mobility Report," https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-november-2016.pdf, 2016, Accessed: October 5, 2017.

[2] ——, "Ericsson Mobility Report – MWC Edition," https://www.ericsson.com/assets/local/mobility-report/documents/2015/ericsson-mobility-report-feb-2015-interim.pdf, 2015, Accessed: October 5, 2017.

[3] S. S. Krishnan and R. K. Sitaraman, "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs," in *ACM Conference on Internet Measurement (IMC)*, 2012, pp. 2001–2014.

[4] J. Summers, T. Brecht, D. Eager, and A. Gutarin, "Characterizing the Workload of a Netflix Streaming Video Server," in *IEEE International Symposium on Workload Characterization (IISWC)*, 2016, pp. 100–111.

[5] C. Koch, G. Krupii, and D. Hausheer, "Proactive Caching of Music Videos based on Audio Features, Mood, and Genre," in *ACM Multimedia Systems (MMSys)*, 2017, pp. 100–111.

[6] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless Video Content Delivery through Distributed Caching Helpers," in *IEEE INFOCOM*, 2012, pp. 1107–1115.

[7] M. Seufert, V. Burger, and T. Hoßfeld, "HORST-Home Router Sharing based on Trust," in *IEEE International Conference on Network and Service Management (CNSM)*, 2013, pp. 402–405.

[8] A. Lareida, G. Petropoulos, V. Burger, M. Seufert, S. Soursos, and B. Stiller, "Augmenting Home Routers for Socially-aware Traffic Management," in *IEEE Conference on Local Computer Networks (LCN)*, 2015, pp. 347–355.

[9] C. Pleşca, V. Charvillat, and W. T. Ooi, "Multimedia Prefetching with Optimal Markovian Policies," *Journal of Network and Computer Applications*, Vol. 69, pp. 40–53, 2016.

[10] A. Gouta, D. Hausheer, A.-M. Kermarrec, C. Koch, Y. Lelouedec, and J. Rückert, "CPSys: A System for Mobile Video Prefetching," in *IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2015, pp. 188–197.

[11] S. Wilk, D. Schreiber, D. Stohr, and W. Effelsberg, "On the Effectiveness of Video Prefetching Relying on Recommender Systems for Mobile Devices," in *IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2016, pp. 429–434.

[12] S. Wilk, J. Rückert, T. Thräm, C. Koch, W. Effelsberg, and D. Hausheer, "The Potential of Social-aware Multimedia Prefetching on Mobile Devices," in *IEEE International Conference and Workshops on Networked Systems (NetSys)*, 2015, pp. 1–5.

[13] M. A. Kaafar, S. Berkovsky, and B. Donnet, "On The Potential of Recommendation Technologies for Efficient Content Delivery Networks," *ACM SIGCOMM Computer Communication Review*, Vol. 43, No. 3, pp. 74–77, 2013.

[14] X. Bai, F. P. Junqueira, and A. Silberstein, "Cache Refreshing for Online Social News Feeds," in *ACM International Conference on Information & Knowledge Management*, 2013, pp. 787–792.

[15] Z. Wang, L. Sun, S. Yang, and W. Zhu, "Prefetching Strategy in Peer-assisted Social Video Streaming," in *ACM International Conference on Multimedia*, 2011, pp. 1233–1236.

[16] Z. Li, H. Shen, H. Wang, G. Liu, and J. Li, "SocialTube: P2P-assisted Video Sharing in Online Social Networks," in *IEEE INFOCOM*, 2012, pp. 2886–2890.

[17] N. Do, Y. Zhao, S.-T. Wang, C.-H. Hsu, and N. Venkatasubramanian, "Optimizing Offline Access to Social Network Content on Mobile Devices," in *IEEE INFOCOM*, 2014, pp. 1950–1958.

[18] C. Koch and D. Hausheer, "Optimizing Mobile Prefetching by Leveraging Usage Patterns and Social Information," in *IEEE International Conference on Network Protocols (ICNP)*, 2014, pp. 293–295.

[19] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Vol. 96, No. 34, 1996, pp. 226–231.

[20] C. Koch, N. Bui, J. Rückert, G. Fioravantti, F. Michelinakis, S. Wilk, J. Widmer, and D. Hausheer, "Demo: Media Download Optimization through Prefetching and Resource Allocation in Mobile Networks," in *ACM Multimedia Systems (MMSys)*, 2015, pp. 85–88.

[21] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[22] X. Cheng and J. Liu, "NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing," in *IEEE INFOCOM*, 2009, pp. 1152–1160.

[23] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A Close Examination of Performance and Power Characteristics of 4G LTE Networks," in *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2012, pp. 225–238.

[24] A. Gouta, D. Hong, A.-M. Kermarrec, and Y. Lelouedec, "HTTP Adaptive Streaming in Mobile Networks: Characteristics and Caching Opportunities," in *IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2013, pp. 90–100.