A Joint Online Transcoding and Delivery Approach for Dynamic Adaptive Streaming

Zhi Wang, Lifeng Sun, Chuan Wu, Wenwu Zhu, *Fellow, IEEE,*
Qidong Zhuang, and Shiqiang Yang, *Senior Member, IEEE*

Abstract—Dynamic adaptive streaming has emerged as a popular approach for video services in today's Internet. To date the two important components in dynamic adaptive streaming, video transcoding that generates the adaptive bitrates of a video and video delivery that streams the videos to users, have been separately studied, resulting in a huge waste of computation and storage resource due to producing and caching different versions of videos regardless of their demands. We conduct extensive measurement studies of video sharing systems, including an IPTV service which streams regular, professionally-made videos and an instant video clip sharing service which provides extremely-short user-generated videos, as well as the availability of computation resource in conventional Content Delivery Networks (CDNs). Based on the measurement insights, we propose an online joint transcoding and delivery approach for adaptive video streaming. We formulate optimization problems to enable high streaming quality for the users, and low computation and replication costs for the system. In particular, our strategy connects video transcoding and video delivery based on users' preferences of CDN regions and regional preferences of video versions. We analyze hardness of these problems and design distributed solutions. Extensive trace-driven experiments further demonstrate the superiority of our design.

1 Introduction

Dynamic adaptive streaming over HTTP (DASH) has emerged as a popular video streaming approach [1], widely implemented and supported by the industry, including Apple HTTP Live Streaming, Microsoft Live Smooth Streaming, and Adobe Adaptive Streaming. It allows users with heterogeneous and dynamically changing network conditions to receive an adaptive bitrate, achieving the best video streaming experience in different contexts [2].

In adaptive streaming, video service providers have to not only deliver the video *segments* (data blocks in a video that can be downloaded over HTTP and played independently), but also transcode the videos to different *versions* (*i.e.*, videos with different bitrates of the same content) for the users. In this paper, we refer to *transcoding* as transcoding a video to different bitrates, which may consume a lot of computation resource [3]. To date, existing approaches separately perform video transcoding and video delivery: being unaware of which segments users will request, they transcode every video published with a set of fixed versions, and replicate segments of different versions using the same strategy.

Problems of the traditional approaches for adaptive streaming are as follows: (1) Prefixed versions only allow users to choose from a small set of candidate bitrates, which cannot effectively adapt to the changing network conditions. (2) To address this problem, video providers increase the number of adaptive versions — as both the number of uploaded videos and the number of versions are increasing, a huge amount of computation resource is required to transcode all the videos to all the versions [4]. As the distribution of video segment popularity is becoming significantly heavy-tailed, *i.e.*, a substantial fraction of video segments are not requested at all, pre-transcoding them could be a huge waste of valuable computation resource. The situation is exacerbated by today's user-generated-content (UGC)-based video sharing services [5]. (3) In addition, traditional approaches are oblivious of users' *preferences* of different *peering servers* (*i.e.*, servers directly uploading the video segments to users) when streaming video segments to users, leading to a mismatch between the download speed and the required segment bitrate, *e.g.*, a user being able to receive a high-bitrate segment might be redirected to a peering server with a slow connection to the user [6].

To address these problems, we propose to jointly schedule segment transcoding and delivery in an online manner, using geo-distributed computation and bandwidth resources. This design philosophy allows us to jointly optimize the streaming quality for users and minimize the computation and bandwidth cost for transcoding and replicating the video segments. To motivate our study, we measure the user request patterns of different video streams in a representative video streaming service in China, BesTV [7] (an IPTV system serving

Z. Wang is with the Graduate School at Shenzhen, Tsinghua University, Shenzhen, China (email: wangzhi@sz.tsinghua.edu.cn).
L. Sun, W. Zhu and S. Yang are with the Department of Computer Science and Technology, Tsinghua University, Beijing Key Laboratory of Networked Multimedia, Beijing, China (e-mail: {sunlf, wwzhu, yangshq}@tsinghua.edu.cn).
C. Wu is with the Department of Computer Science, the University of Hong Kong, Hong Kong (e-mail: cwu@cs.hku.hk).
Q. Zhuang was with Tencent when this work was done, and is now with The Hong Kong University of Science and Technology, Hong Kong (e-mail: qzhuang@ust.hk).

over 16 million users) and Weishi [8] (an instant video clip sharing service serving over 18 million users). We have made the following observations: (**i**) due to the skewness of the distributions of popularity of the videos, segments and versions, the online transcoding paradigm is promising to significantly reduce the demand for computation resource.

To exploit the practical feasibility of implementing joint transcoding and delivery in widely deployed CDNs, we further measure the availability of computation and bandwidth resources in Tencent CDN [9], which serves over 70% of the traffic from one of the largest content providers in China. We have further made the following observations: (**ii**) A substantial amount of idle computation resource is available on the *backend servers* (*i.e.*, servers supporting the peering servers), and the amount is relatively stable over time, indicating that online transcoding can be effectively performed by these backend servers. (**iii**) Peering servers are deployed at different geographic locations and connected to different Internet Service Providers (ISPs), such that users' download speeds differ when streaming from servers in different CDN regions. Therefore, users have different preferences of CDN servers in different regions from which to receive segments, while servers in different regions have different preferences of versions of video segments to transcode.

Based on our previous work [10], we substantially extend the measurement study, the analysis and evaluation in this paper. Our contributions are summarized as follows:

▷ We conduct large-scale measurement studies not only on traditional adaptive video streaming services, but also on the recent instant social video sharing services, to motivate our proposal, to derive guidelines for our design, and to investigate the feasibility of its practical implementation.

▷ To achieve good streaming qualities, low computation resource consumption, and low segment replication cost, we connect video transcoding and video delivery based on users' region preferences and regional version preferences, *i.e.*, we use users' preferences of regions to redirect them to their ideal peering servers, and use the regional version preferences to schedule the transcoding tasks. In particular, segments are transcoded to a set of pre-defined versions according to the strategy designed, in the on-demand fashion. We formulate optimization problems for video transcoding/delivery decisions, analyze the NP-hardness of the problems and design practical algorithms to compute the solutions.

▷ We conduct extensive trace-driven experiments to evaluate the effectiveness of our algorithms, and show that both users' experience and system's computation resource utilization are improved by our proposal.

The rest of the paper is organized as follows. We discuss related works in Sec. 2. We present the measurement insights that motivate our design in Sec. 3, present our detailed design in Sec. 4, and verify its effectiveness by experiments in Sec. 5. Finally we conclude the paper in Sec. 6.

2 Related Work

We survey related literature on today's HTTP-based adaptive streaming, quality of experience in adaptive streaming, CDN-based streaming architectures, and conventional video transcoding schemes.

2.1 HTTP-based Adaptive Streaming

The rapid growth of HTTP streaming is partly due to the extensive support from content distribution networks (CDN) [11]. HTTP video streaming works by breaking the overall video stream into a sequence of small HTTP-based file downloads. Due to the best-effort nature of streaming videos over the Internet, Dynamic Adaptive Streaming over HTTP (DASH) has been proposed to adapt the streaming rates from Web servers. DASH was developed in 2010 [1] and has become a new standard in 2011 [2] to enable high-quality streaming of media content over the Internet, delivered from conventional HTTP Web servers. However, in the conventional DASH framework, how the segments should be jointly transcoded and delivered is not technically discussed.

2.2 Quality of Experience in Adaptive Streaming

The quality of experience (QoE) in video streaming is a critical factor that reflects the effectiveness of the streaming strategies. For conventional video streaming, metrics including bitrate, packet loss and delay are generally used [12], [13]. For adaptive video streaming, since bitrate changes during a session, the inference of the quality of experience has to take the new feature into consideration. Mok et al. [14] studied the inference of the quality of experience in adaptive video streaming such as DASH, and observed that both streaming bitrate and user activities can affect the quality of experience. In our design and evaluation, we take these metrics of quality of experience into account.

2.3 Streaming Architecture based on CDN

Many architectures have been proposed to implement large-scale video streaming services including the CDN-based architectures [11]. CDNs can significantly assist in HTTP-based streaming with servers deployed in multiple geographical locations across multiple ISPs [15]. Since today's CDNs are mainly designed and optimized to serve web contents [16], HTTP video streaming can be regarded as downloading video segments progressively from web servers via the HTTP protocol. Users experience higher-quality streaming by receiving streams at more reliable bandwidth from the CDN servers. Recently, Adhikari *et al.* [6] proposed a multi-CDN scheme for real-world video systems to further improve the streaming quality. Traditional studies on video streaming have been focusing on improving the connectivity between streaming servers and users from the network aspect.

2.4 Video Transcoding Schemes

Compared with the traditional video streaming paradigm, DASH enables a much larger number of quality versions, requiring a huge amount of computation resource to transcode these versions of videos. Dedicated transcoders are developed to speed up video transcoding [17]. There were also some works on using the computation resource in a cloud cluster for video transcoding. Lao *et al.* [4] designed a MapReduce-based video transcoding scheme for distributing transcoding tasks. Huang *et al.* [18] proposed CloudStream, which schedules the video transcoding tasks inside a cluster according to properties of the videos. Traditional studies on video transcoding have been exploiting dedicated devices or computation resource, leading to the decoupling of segment transcoding and delivery.

Most related works on adaptive streaming have investigated video delivery and video transcoding separately, *i.e.*, videos are pre-transcoded centrally, and then replicated to CDN servers for delivery using the same strategy, *e.g.*, a full replication scheme. In this paper, we explore the design space of joint transcoding and delivery using geo-distributed computation and network resources.

3 Measurements and Observations

We conduct measurement studies to motivate our design, and summarize design principles learnt.

3.1 Measurement Setup

To demonstrate the benefits and feasibility of our proposal, we use large-scale measurement studies based on valuable traces collected from BesTV and Tencent CDN.

3.1.1 Traces of Users' Video Viewing Patterns

To study the potential of using an online transcoding scheme to save computation resource, we have collected real-world traces on video access patterns from two respective video services: (1) BesTV, an Internet Protocol Television (IPTV) system, and (2) Weishi, an instant social video sharing system, both based in China. Detailed traces on user behaviors in both systems are collected as follows.

▷ In BesTV, videos are published into 17 categories, and pre-transcoded into 4 versions (700 Kbps, 1300 Kbps, 2300 Kbps and 4000 Kbps). We collected viewing activities of users in Heilongjiang province in November 2012, about how 190K videos were watched by users from over 3 million IP addresses. For each of the videos, the traces record which segments were downloaded by which users, including the time stamp when a segment was downloaded, the user ID, the video ID, the size and version of the segment, and the time spent in downloading the segment. Using these data, we can show the great potential of our joint transcoding and streaming paradigm in Sec. 3.2.

▷ In Weishi, extremely short videos (in 10 seconds) are generated by individuals and shared with their "followers". Each video in Weishi is transcoded into the following versions: a) 480x480, 2000 Kbps; b) 480x480, 1050 Kbps; c) 480x480, 500 Kbps; d) 480x480, 300 Kbps. In the Weishi traces, each item records when a video with a particular version is downloaded by which IP, and from which CDN server. Using these traces, we are able to demonstrate the effectiveness of online transcoding for instant video sharing services.

3.1.2 Traces of CDN Characteristics

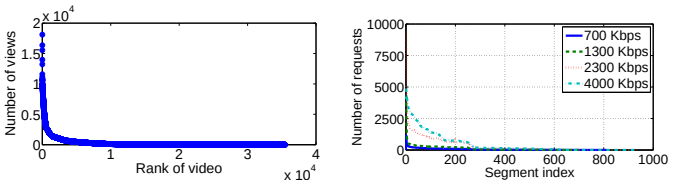
To study the feasibility of online transcoding in a CDN system, which has already been widely used for adaptive streaming, we collected traces of the backend and peering servers from Tencent CDN, as follows. (1) *CPU load patterns*: To study the computation resource availability for segment transcoding, we collected the CPU load traces from the backend servers. In particular, the average CPU load of over 5 thousand servers was recorded every 5 minutes, throughout the whole month of March 2013. Each CPU load trace item contains the timestamp and the CPU load, recorded as the average number of processes waiting on each CPU core, *e.g.*, a CPU load greater than 1 indicates that the server is fully loaded. (2) *Bandwidth patterns*: To study the users' preferences of CDN regions, and the regional preferences of versions to transcode, we have collected traces including 3.39 billion TCP connections from peering servers located at 55 regions in May 2013. These TCP connections were established to download contents with sizes varying from tens of bytes to 4.8 GB. Each of the trace items contains the following information: the timestamp indicating when a TCP connection was established, the client IP, the number of downloaded bytes and the duration of the connection. In Sec. 3.3, we use these data to study the feasibility and provide guidelines for our design.

3.2 Users' Watching Behavior

3.2.1 Popularity of Professional Videos Published in the IPTV Service

Based on the video viewing records in BesTV, Fig. 1(a) illustrates the distribution of video popularity. Each sample represents the number of user requests for a video in one month versus the rank of the video. We observe that over 53% of these videos had no viewers in a month. This can be explained by the fact that the time users spend in watching videos grows much slower than the growth of the number of videos, and such skewness of the popularity distribution is also prevalent in other UGC-based video sharing systems, such as YouTube [5].

In Fig. 1(a), only 13% of the videos have a monthly viewing number larger than 500. We further investigate how different segments with different versions inside such a relatively popular video (with about 1,000 segments), were requested by users. Fig. 1(b) illustrates the distribution of the requests for segments of one of the



(a) Number of user views versus the rank of video.

(b) Number of segment requests versus segment index for different versions.

Fig. 1. Popularity of videos, segments and versions in BesTV (Heilongjiang, November 2012).

most popular videos. Each curve represents the number of segment requests versus the index of the segment. We observe that (1) only a small range of segments were requested by many users, *e.g.*, the first tens of the segments; (2) different versions received different numbers of requests, *e.g.*, the version of 4000 Kbps was requested by much more users; and (3) a large fraction of segments were requested by no one for some versions, *e.g.*, the last segments of the 700 Kbps and 1300 Kbps versions.

3.2.2 *Viewing Patterns in Instant Social Video Sharing Services*

We first study the distribution of the popularity of instant video clips generated by users in Weishi. As illustrated in Fig. 2, a sample represents the number of views of a video versus the rank of the video in one day. We have sampled over $10,000$ videos from over 1.6 million videos viewed on February 27, 2014. We observe that the popularity distribution can be nicely fit using a zipf distribution (with the space parameter $s = 1.209$), indicating that viewers in the instant video sharing service also concentrate on the most popular videos in a highly skewed manner. Uniformly transcoding every video is thus a waste of computation resource.

Next, we investigate how videos are generated by users in the instant video sharing service. In Fig. 3, the two curves represent the number of video uploads and the number of video requests by users in each hour. Our observations are: (1) The number of requests is larger than the number of uploads by 250 times. Over time, there will be much more videos generated than what users may watch. (2) The peak hour of video requests is several hours later than that of video uploads: the time difference may allow us to perform the transcoding task schedule, detailed in Sec. 4.

We then study how the user-uploaded videos were requested in the following time window after they were generated. In Fig. 4, the red reference line denotes the number of videos uploaded by users in one day (February 26, 2014), and each bar represents the number of videos among them that are requested (for the first time) on the following xth day ($x = 1, 2, 3$). We observe that around 70% of the videos were requested in the same day when they were uploaded, but much fewer attract viewers after that. Around 30% of user-generated

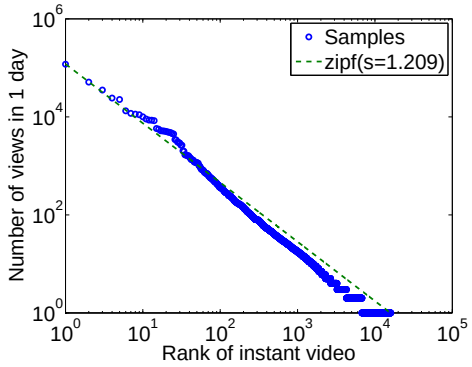


Fig. 2. Number of views versus the rank of video in Weishi (February 27, 2014).

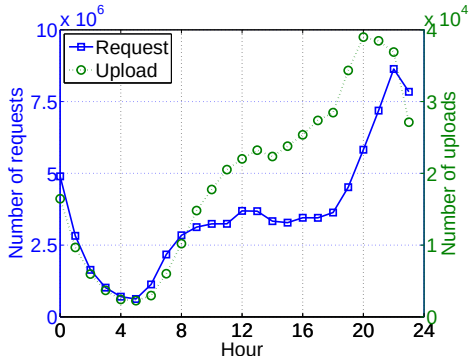


Fig. 3. Number of video requests and video uploads over time in Weishi (February 27, 2014).

videos were not viewed at all by others after they were published.

Finally, we study when the videos (that were ever requested) were requested by their first viewers. We plot the CDF of the elapse between the upload time of a video and the time the first request for the video was issued in Fig. 5, over $300,000$ videos. We observe that over 40% (resp. 55% and 85%) of the videos were requested 1 hour (resp. 8 hours and 24 hours) after they were uploaded. These observations show that (1) pre-transcoding which transcodes videos that are not needed in the near future wastes computation resource, and (2) the upload-download elapse of videos is different, resulting in the different "urgency levels" for them to be transcoded.

The above observations indicate that as more and more videos are published, by both the professional content providers and individuals, pre-transcoding every segment of all videos into an increasing number of versions can be a huge waste of computation resource, motivating our online transcoding solution.

3.3 Availability of Idle Computation Resource in CDNs

Realtime monitoring of Akamai's severs [19] has revealed that the CPU load on their CDN servers can be efficiently measured, which varies across different servers.

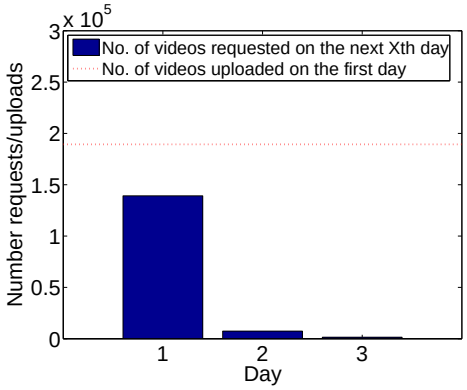


Fig. 4. Number of videos requested by users in the first three days after they were generated.

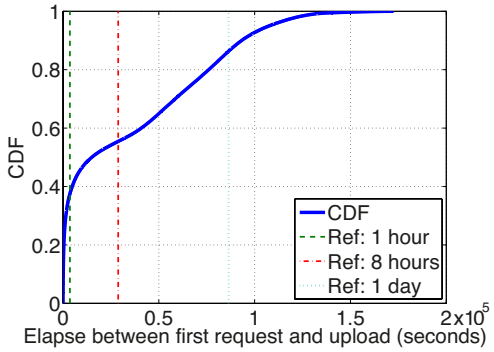
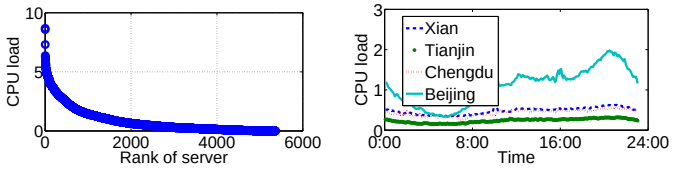


Fig. 5. CDF of elapse between upload and the first request.

We further measure the idle computation resource on Tencent CDN servers, to explore the feasibility of performing video transcoding using such idle resource.

Fig. 6(a) plots the CPU load — the average number of processes waiting on each CPU core of the backend servers — in a representative time slot of 15 minutes. We observe that in this time slot, the CPU load of the $5,441$ backend servers varied from around 0 to 8.6, and as many as 72.4% (*resp.* 55.9%) of the backend servers had a CPU load smaller than 1.0 (*resp.* 0.5), showing that a substantial amount of the computation resource in the CDN is available and can be exploited. The reason for such high availability is that many backend servers are only assigned to simple I/O tasks, *e.g.*, loading data from the distributed storage system for the peering servers.

We further study the availability of computation resource in a region of the CDN. We use a *city-ISP* pair to identify a region. Fig. 6(b) illustrates the CPU load on four largest regions (Xian-China Telecom, Tianjin-China Telecom, Chengdu-China Telecom and Beijing-China Telecom) that Tencent CDN covers, *i.e.*, the average CPU load of all the backend servers in that region. We also observe the existence of available computation resource at the region level, and the variance of the CPU load in different regions, *e.g.*, the CPU load of Xian-China Telecom is much lower than that of Beijing-China Telecom.



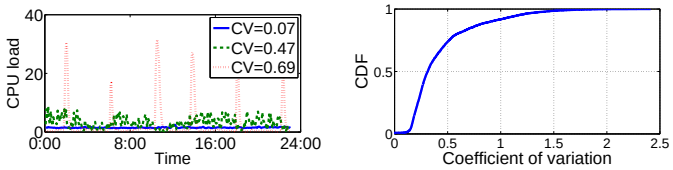
(a) Average CPU load in 15 minutes versus the rank of the server (8PM, March 5, 2013).

(b) Average CPU load on servers of different regions (March 5, 2013).

Fig. 6. Average CPU load of backend servers.

Since these back-end servers can be scheduled to run different tasks, their idle computation resource may vary over time. To make use of these resource for segment transcoding, relative stability of the idle resource which reflects the churning level of CPU loads of servers overtime, is a key. Fig. 7(a) plots the CPU load of 3 servers, which is sampled once every 5 minutes. We calculate an average coefficient of variation (CV) to evaluate the daily churning level of the CPU load on each server, as follows: $CV = 1/24 \sum_{h=0}^{23} \sqrt{E[(X_h - \bar{X_h})^2]}/\bar{X_h}$, where random variable X_h denotes the CPU load in hour h (expectation computed using 12 samples in each hour). A large CV implies more churns of the CPU load over time. We observe that the server with a relatively stable CPU load over time achieves $CV = 0.07$, and the server with significantly varying CPU load achieves $CV = 0.69$.

Fig. 7(b) further plots the distribution of CVs of all the backend servers. We observe that over 70% of the servers achieve a CV smaller than 0.5, indicating that the CPU load of many backend servers is relatively stable, such that their capacities for performing video transcoding in the near future can be guaranteed. We seek to exploit such resource availability in our transcoding task scheduling.



(a) CPU load on three representative back-end servers (March 5, 2013).

(b) CDF of the coefficient of variation of all the servers (March 5, 2013).

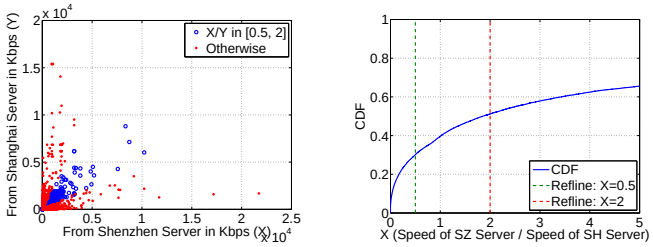
Fig. 7. Variation of server CPU load over time.

3.4 Users' Preferences of Streaming from Different CDN Regions

In the context of online transcoding, we are allowed the degree of freedom that segments can be transcoded in different CDN regions. Next, we explore the guidelines for such transcoding schedule.

We show the diversity in download speed across regional CDN servers. Which version of a video segment is requested by a user depends on the user's download

speed. Based on the TCP traces of the peering servers, we compare the download speeds of about 150 users who downloaded from different peering servers in the same 10 minutes on May 4, 2013. In Fig. 8(a), each sample is the average download speed of a user downloading from a regional CDN server deployed in Shanghai (SH), versus the average download speed of the same user downloading from another server deployed in Shenzhen (SZ), both with the same ISP. We further plot the CDF of the ratio of the two speeds in Fig. 8(b), and observe that for over 79% of the users, their download speeds differs by > 2 times when they are downloading from different servers (marked red in this figure), indicating that redirecting users to their ideal peering servers can help a majority of users to receive a better streaming quality.



(a) Comparison of average download speeds of users downloading from different peering servers (May 4, 2013).

(b) CDF of ratio of two speeds.

Fig. 8. Download speeds of users downloading from different peering servers.

3.5 Regions' Preferences of Different Versions to Transcode

On the other hand, to study the regional preference of versions to transcode, we calculate the average download speeds of users downloading from the peering servers. In Fig. 9, each sample represents the average download speed in one day, of all users served by a server versus the rank of the server. We observe that the average download speed varies quite significantly across these peering servers, from 170 Kbps to 1.1 Mbps.

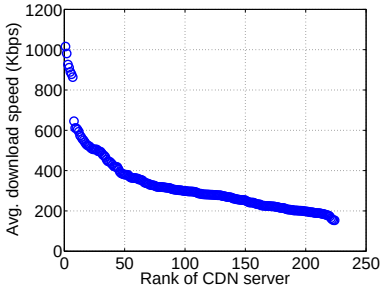


Fig. 9. Average download speed of users from different peering servers on May 4, 2013.

We further study the download speeds from servers in different regions to users. In Fig. 10, each bar represents the minimal, average and maximal download speeds from the peering servers in a region. We observe that the average download speeds across different regions vary from 180 Kbps (region BJT, *i.e.*, Beijing-China Telecom) to 512 Kbps (region ZJM, *i.e.*, Zhejiang-China Mobile).

These observations tell us the following: Peering servers are physically deployed at different locations and with different ISPs, so that the Internet connectivity and the average bandwidth capacity are different. Given the different connectivities, servers in different regions are suitable to transcode segments into different versions, *e.g.*, a region with a low server-to-user download speed should produce low-bitrate segments. Satisfying such preferences of regions for transcoding task schedule can lead to reduced *cost* of replicating transcoded segments, since segments are transcoded where they are to be requested.

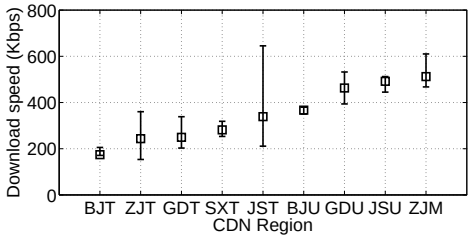


Fig. 10. Average user download speed in different CDN regions on May 4, 2013.

4 Joint Online Transcoding and Geo-Distributed Delivery

In this section, we present our design of joint online transcoding and geo-distributed delivery strategy. Before presenting the details, we give the assumptions/non-goals in our study: first, we assume the transcoding is performed from high-bitrate versions to low-bitrate versions; second, we assume segments of the highest versions are already replicated to the geo-distributed CDN regions; third, we focus on improving users' streaming quality and reducing the computation resource consumption, while not particularly considering the allocation of storage resource.

4.1 Framework

Fig. 11 illustrates our design, where segments in different versions of videos are transcoded upon users' requests. In this example, $s1, s2, s3, s4$ represent segments of different versions, requested by a user during her streaming session. $R1$, $R2$ and $R3$ are CDN *regions* (each is represented by a pair of a geographical location and an ISP) where backend servers and peering servers are deployed. Segments can be transcoded at selected regions (*e.g.*, $s2$ is transcoded in region $R1$), replicated between regions (*e.g.*, $s1$ is replicated from region $R3$ to $R1$), and delivered to users, all in an online manner.

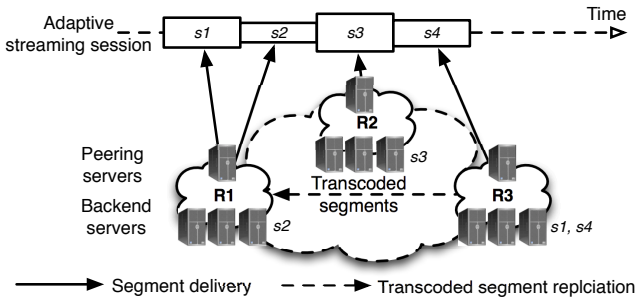


Fig. 11. Joint online transcoding and geo-distributed streaming: an illustration

Fig. 12 further illustrates the framework of our online transcoding and delivery scheme, which schedules segment transcoding and replication periodically: based on the recent information collected in time slot $T-1$, we perform transcoding and replication of segments that are likely to be requested in time slot T. In practice, the length of a time slot should be similar to the playback duration of a segment. In our experiments later, set the length of a time slot to be 10 seconds, the same as the playback duration of a segment. The collected information includes: (1) Users' preferences of different regions to receive segments. In our design, we allow users to use a bandwidth estimation approach (*e.g.*, abget [20] which incurs little bandwidth overhead to measure the end-to-end bandwidth) to rank a set of candidate peering servers, in the descending order of the estimated download speed. (2) The number of requests for a particular segment, which can be predicted according to users' segment requests in previous time slots, *e.g.*, we assume most of users will play videos in a consecutive way, issuing very few seeks during the playback [21], and we predict the number of users requesting a segment accordingly as the number of users downloading the previous segment in the previous time slot. Based on the CDN-to-user bandwidth information in the previous time slot, we further estimate which particular versions of the segments will be requested by users in the next time slot. (3) The idle computation resource. As many backend servers have stable CPU load over time according to our measurement study, we use the level of computation resource in the current time slot as the available computation resource in the next time slot. Other regression models (*e.g.*, ARIMA [22]) can also be explored to achieve better prediction accuracy, which we will investigate in the future.

Using such information, we perform the following: (1) *User redirection*. To enable high-quality streaming, our design redirects users to their ideal regions so that they can receive segments at high bitrates. We are redirecting users at regional level, *i.e.*, a region with the highest CDN-to-user bandwidth will be selected to serve a user's request, and peering servers in the same region will serve user requests in a round-robin manner. (2) *Transcoding segment selection*. Backend servers with idle CPU resource perform video transcoding, by slicing a video into multiple segments, containing closed groups of pictures (GoPs) that can be transcoded independently [18]. To allow a smooth playback, when a segment of a particular version is not transcoded timely, we send the requesting user the segment of an alternative version, whose bitrate is the closest lower to the requested version. We prioritize requested versions of segments that are less desirable to be replaced by alternative versions due to large bitrate difference between them, to be transcoded when computation resource is not sufficient. (3) *Transcoding task assignment*. A transcoded segment is cached by the backend servers and replicated to other regions, according to our replication strategy. Transcoding is performed at strategically selected regions, so that the cost of replicating the transcoded segments to other regions can be minimized.

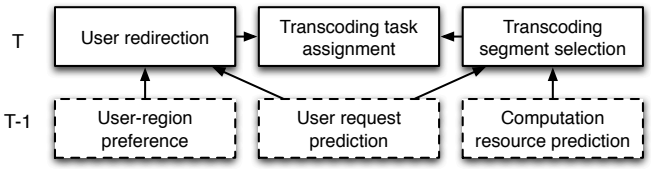


Fig. 12. Framework of our joint online transcoding and delivery mechanism.

4.2 Quality-Driven Redirection

In our design, a user is redirected to her ideal region where segments are generated on the fly. This design principle allows users to choose a CDN region with the largest download speed to receive the segments, without considering the segment availability.

We formulate the region-level user redirection problem as an optimization problem. Let $\mathbf{U}^{(T)}$ denote the predicted set of users requesting different segments in the system in time slot T, and $\mathbf{R}$ be the set of CDN regions. We use $D^{(T)}$ to denote a redirection strategy, where the binary variable $D^{(T)}_{u,r} = 1$ (*resp.* 0) indicates that user u will (*resp.* will not) be downloading from region r in the next time slot T.

In the context of adaptive video streaming, we assume that users expect to receive a large bitrate for good streaming quality whenever possible. Thus, we use $H_{u,r}$ to denote user u's preference to download from region r. $H_{u,r}$ can be defined as a concave increasing function of the estimated download speed achieved when user u downloads from peering servers in region r. The optimization problem is as follows:

$$\max_{D^{(T)}} \sum_{u \in \mathbf{U}^{(T)}, r \in \mathbf{R}} H_{u,r} D^{(T)}_{u,r}, \tag{1}$$

subject to:

$$\sum_{r\in\mathbf{R}} D_{u,r}^{(T)} \leq 1, \forall u \in \mathbf{U}^{(T)},$$
$$\sum_{u\in\mathbf{U}^{(T)}} D_{u,r}^{(T)} B_{L_{u,r}} \leq W_r, \forall r \in \mathbf{R},$$
$$D_{u,r}^{(T)} \in \{0,1\}, \forall u \in \mathbf{U}^{(T)}, r \in \mathbf{R},$$

where $L_{u,r}$ is the version with the highest bitrate that u can receive when she downloads from region r, B_v is the bitrate of version v, and W_r is the bandwidth capacity of peering severs in region r. The rationale of the optimization is to maximize the streaming quality for users by the redirection.

Theorem 1. *The problem of redirecting users to CDN regions such that their preferences can be maximally satisfied, as formulated in (1), is NP-hard.*

Proof: We reduce a conventional 0/1 knapsack problem, which is NP-hard, to this problem. The 0/1 knapsack problem has the following structure:

$$\max \sum_{i=1}^{n} v_i x_i,$$

subject to

$$\sum_{i=1}^{n} \alpha_i x_i \leq \beta,$$
$$x_i \in \{0,1\},$$

where $x_i, i = 1,2,\ldots,n$ are the optimization variables. For any 0/1 knapsack problem as above, we reduce it to our redirection problem as follows:

1) Let $\mathbf{U}^{(T)} = \{1,2,\ldots,n\}$, $\mathbf{R} = \{1\}$;
2) Let $H_{i,1} = v_i, i = 1,2,\ldots,n$;
3) Let $B_{L_{i,1}} = \alpha_i, i = 1,2,\ldots,n$;
4) Let $W_1 = \beta$.

According to the procedure above, these reduction operations take linear time to complete, and the final results for the 0/1 knapsack problem are $x_i = D_{i,1}^{(T)}, i = 1,2,\ldots,n$. Therefore, our problem is NP-hard. □

We design an algorithm to heuristically solve the optimal user redirection problem in a distributed manner: (1) When a user starts to watch a video, the system assigns her a list of candidate peering servers from regions with the lowest load. (2) The user ranks these servers in descending order of the estimated download speeds as discussed in Sec. 4.1, and sends connection requests to these servers. (3) On the other hand, a peering server may receive connection requests from many users, and can only accept a limited number of users according to its available bandwidth W_r. The request from user u is prioritized to be accepted if she has a larger $H_{u,r}/B_{L_{u,r}}$ with the CDN region r — this value reflects a marginal "gain" in streaming quality by a unit of bandwidth allocated. (4) The user selects the best peering server from the ones accepting her request according to the ranked list. In a real system, this algorithm can be effectively implemented and executed in a distributed manner.

4.3 Region-Preference-Aware Transcoding Schedule

After users are redirected to the CDN regions, they send requests for video segments of different versions. Based on the segment request prediction, we perform the transcoding task schedule, which works in two steps: (1) we prioritize the segment transcoding tasks such that important segments are transcoded more urgently; and (2) we distribute the transcoding tasks to CDN regions, such that segments are transcoded where they are more likely to be requested.

4.3.1 Prioritizing Segment Transcoding Tasks

We prioritize the segment transcoding tasks according to the importance of these segments. We denote $e_{(s,v)}^{(T)}$ as the importance level of segment s of version v in time slot T, which depends on the following factors: (1) the estimated number of user requests for the segment, discussed in Sec. 4.1; and (2) the quality-wise importance of the segment, which depends on the existing versions of the same segment. In particular, $e_{(s,v)}^{(T)}$ can be calculated as follows:

$$e_{(s,v)}^{(T)} = Q_{(s,v)}^{(T)} Y_{(s,v)}^{(T)},$$

where $Q_{(s,v)}^{(T)}$ denotes the predicted number of requests of the particular segment (s,v) in the next time slot T, and $Y_{(s,v)}^{(T)}$ is the streaming quality gain if the segment is transcoded to version v. $Y_{(s,v)}^{(T)}$ is calculated as the "mismatch" level of the bitrate if v is not transcoded as follows:

$$Y_{s,v}^{(T)} = \begin{cases} \min_w (B_v - B_w)/B_v, & \exists w \in \mathbf{G}_s^{(T)}, w < v \\ 1, & \text{otherwise} \end{cases},$$

where $\mathbf{G}_s^{(T)}$ is the set of all the versions of the segment existing in the system. $Y_{s,v}^{(T)}$ is in the range $(0,1]$. When there exist replacement versions with lower bitrates, the one with the closest bitrate will be served to users, yielding $\min_w (B_v - B_w)/B_v$, where B_v is the original bitrate and B_w is the closest bitrate; when there is no replacement version, $Y_{s,v}^{(T)}$ is set to 1. A larger $Y_{s,v}^{(T)}$ thus indicates that users will receive a highly mismatched bitrate if the version v is not transcoded, and version v is important to segment s in terms of streaming quality of its receivers. Fig. 13 gives an example of the importance of segments: a solid block represents a segment transcoded, and a dashed block represents one that is not transcoded yet. When segments $(s1,v3)$ and $(s2,v3)$ are both requested by the same number of users, $(s1,v3)$ is prioritized to be transcoded over $(s2,v3)$, since users requesting $(s2,v3)$ can be served by an alternative version $(s2,v2)$, while no version of segment $s1$ exists in the system.

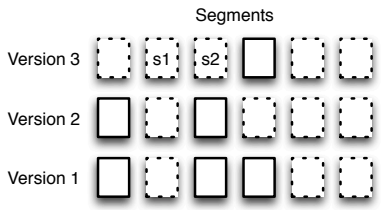


Fig. 13. Segment importance: an illustration.

Based on the definition of the importance level of segments, we determine which segments among all the requested ones in $\mathbf{K}^{(T)}$, to be transcoded by formulating it as an optimization problem, as follows:

$$\max_{P^{(T)}} \sum_{(s,v)\in\mathbf{K}^{(T)}} P^{(T)}_{(s,v)} e^{(T)}_{(s,v)}, \tag{2}$$

subject to:

$$\sum_{(s,v)\in\mathbf{K}^{(T)}} P^{(T)}_{(s,v)} C_{(s,v)} \leq \sum_{r} I^{(T)}_{r},$$

$$P^{(T)}_{(s,v)} \in \{0,1\}, \forall (s,v) \in \mathbf{K}^{(T)},$$

where $P^{(T)}_{(s,v)} = 1$ (*resp.* 0) indicates that segment (s,v) will (*resp.* will not) be transcoded in the next time slot T, $C_{(s,v)}$ is the amount of computation resource required to transcode a segment (s,v), and $I^{(T)}_{r}$ is the aggregated idle computation resource from the CDN region r. According to [18], it takes different CPU times to generate different segments in the same video. In our design, we use the average CPU time spent on generating historical segments of a particular version and size, to estimate the computation resource required to transcode any segment with that version and size.

The optimization, which is a 0-1 knapsack problem, is to select a set of segments that are most important in the next time slot T. We design the following algorithm to solve this problem: (1) We collect the information for prediction in a centralized manner, *e.g.*, users (*resp.* backend servers) report which segments they are downloading (*resp.* the CPU load information) to a centralized server, which will perform the prediction; (2) Based on the prediction, we rank the requested segments in descending order of $e^{(T)}_{(s,v)}/C_{(s,v)}$; (3) We iteratively select segments from the ranked list to transcode, and update computation resource consumption, until the available idle computation resource is used up.

4.3.2 Scheduling Transcoding Tasks across Regions

After the tasks are selected, they are to be scheduled to different regions where backend servers can provide the computation resource. Without loss of generality, we let $A^{(T)}_{(s,v),r} = 1$ (resp. 0) if region r will (resp. will not) transcode segment (s,v) — the segment will be replicated from this region which originally stores the transcoded version, to other regions where users request it.

According to our measurement studies in Sec. 3, heterogeneous preferences of video versions exist at different regions, due to the different download speeds from the servers at different regions. As a result, it is promising to strategically assign transcoding tasks of different segments to backend servers at different CDN regions for a minimized replication cost.

We use $F_{(s,v),r}$ to denote the overall replication cost if segment (s,v) is transcoded in region r. It can be calculated as follows:

$$F_{(s,v),r} = \sum_{r'\neq r, J^{(T)}_{(s,v),r'} > \beta} Z_{r,r'}(s,v),$$

where $J^{(T)}_{(s,v),r'}$ is the number of requests of segment (s,v) to be served by a region r', $Z_{r,r'}(s,v)$ represents the replication cost when segment (s,v) is replicated from region r to region r', depending on the size of the segment and the bandwidth between CDN regions r and r'. Important factors that affect the replication cost are as follows: (1) The pricing strategy for data transmission between different peering points, *e.g.*, Internet service providers may charge differently for traffic across different regions; (2) The load of the peering servers, *e.g.*, a video service provider may want to reserve bandwidth on the heavy-loaded servers to directly serve users instead of replicating segments. β is a threshold of the number of requesting users from a region to trigger a replication. $J^{(T)}_{(s,v),r'}$ can be derived from the optimization in (1), which determines the redirection of users. The rationale of this definition is that, in our design, a transcoded segment can be replicated from where it is transcoded to other regions where it is substantially requested (*i.e.*, $J^{(T)}_{(s,v),r'} > \beta$) — a large $F_{(s,v),r}$ indicates a large replication cost between CDN regions if segment (s,v) is transcoded by region r.

Let $\mathbf{E}^{(T)}$ denote the set of segments to be transcoded in time slot T, calculated according to the segment selection procedure above. The task assignment problem is then formulated as follows:

$$\min_{A^{(T)}} \sum_{(s,v)\in\mathbf{E}^{(T)}} \sum_{r\in\mathbf{R}} A^{(T)}_{(s,v),r} F_{(s,v),r}, \tag{3}$$

subject to:

$$A^{(T)}_{(s,v),r} \in \{0,1\}, \forall (s,v) \in \mathbf{E}^{(T)}, r \in \mathbf{R}$$

$$\sum_{r\in\mathbf{R}} A^{(T)}_{(s,v),r} = 1, \forall (s,v) \in \mathbf{E}^{(T)}$$

$$\sum_{(s,v)\in\mathbf{E}^{(T)}} A^{(T)}_{(s,v),r} C_{(s,v)} \leq I^{(T)}_{r}, \forall r \in \mathbf{R}.$$

The rationale of the optimization is to schedule the segment transcoding tasks to different CDN regions, so that the overall replication cost can be minimized. In our implementation, we design the following algorithm to solve the problem, as follows: (1) We first rank all the pairs of the CDN regions and segments (*i.e.*, $|\mathbf{R}| \times \left|\mathbf{E}^{(T)}\right|$ elements), in ascending order of $F_{(s,v),r}$; (2) we pick the

Algorithm 1 Transcoding task schedule.

procedure TRANSCODING TASK SCHEDULE
  Let $M_r = I_r^{(T)}, r \in \mathbf{R}$
  Let $A_{(s,v),r}^{(T)} = 0, \forall (s,v) \in \mathbf{E}^{(T)}, r \in \mathbf{R}$
  Rank CDN region and segment pairs $(r-(s,v))$ in ascending order of $F_{(s,v),r}$
  for $\forall r-(s,v)$ in the ranked list **do**
    if $C_{(s,v)} \leq M_r$ **then**
      Let $M_r = M_r - C_{(s,v)}$
      Let $A_{(s,v),r}^{(T)} = 1$
      Remove pairs with (s,v) from the ranked list
    end if
  end for
end procedure

region-segment pair, $r-(s,v)$, with the smallest $F_{(s,v),r}$ and assign the transcoding task of segment (s,v) to region r; (3) we update the available computation resource of the selected region, and iteratively perform (2) until all computation resources in all the regions are fully used up. This algorithm can be implemented in a centralized manner, where a central server is deployed to collect the request information from streaming servers and make the decisions. The time complexity of the algorithm is determined by the sort operation to the paris of CDN regions and segments, *i.e.*, $N \log N, N = |\mathbf{R}||\mathbf{E}^{(T)}|$. Since we run the algorithm in each time slot, the number of requested segments is limited; and the number of CDN regions is a constant number. Such implementation has been well applied in peer-assisted on-demand streaming systems [23], where a central server tracks the storage status of peers to help them find each other.

In our design, regions with a request number of a segment larger than β will serve a replication of the segment; while for other regions with numbers of requests smaller than β, they will further redirect the users to other regions with the segment transcoded or replicated, according to the users' preferences.

5 PERFORMANCE EVALUATION

5.1 Experiment Setup

We develop an event-driven simulation platform which takes users' viewing activities, and the transcoding and redirection decisions as events to drive the experiments. We set up our experiments following traces we have collected, and compare our design with a pre-transcoding baseline scheme. Details are as follows.

▷ *Users*. According to models summarized from user viewing traces in BesTV, we simulate $10,000$ users, each of whom repeatedly joins different video sessions. A user is randomly associated with a region among 30 regions, and has different download speeds from different regional CDN servers, based on the region-to-region speeds from our collected TCP traces, to be detailed later. After a user joins the system, she selects a video to watch according to the video popularity distribution, and the probability for a video to be selected is proportional to its popularity, counted based on the number of views in our traces. The indices of the first segments users start to view also follow a Zipf distribution, with a shape parameter 1.29. When playing a video, a user plays (downloads) sequentially the segments, and may jump to a random segment ahead with a probability of 0.05. The rationale is that in a video session, how users request segments follows a pattern that users generally play forward and issue a few seeks, most of which are forward seeks [24]. Before leaving a video session, the number of segments a user downloads follows a Zipf distribution with a shape parameter 1.12.

▷ *Video Provider*. New videos are published every $10,000$ time slots, as professional videos (e.g., movies and TV shows in BesTV) are generally published regularly on a daily basis. The popularity of videos follows a zipf distribution with a shape parameter 1.76. In our experiments, the default number of segments in each video is 200, and the default number of versions is 4, if not specified otherwise. The bitrates of the versions are uniformly distributed between the lowest user download speed, and the highest user download speed. Each segment renders a 10-second playback, and the computation resource required to transcode a segment is randomly distributed within $[5, 10]$ CPU seconds [25].

▷ *CDN Regions*. We simulate 30 regions. We set a region-to-user average download speed according to the download speed of $10,000$ IP prefixes randomly selected from the CDN traces, *i.e.*, the download speed of an IP prefix is the average download speed of users with the same prefix in an one-week time span, varying from 70 Kbps to 2.2 Mbps. In our experiments, the aggregated CDN bandwidth is sufficient for all the users to stream at their ideal bitrates, and we randomly divide the bandwidth allocation across the regions. We assign the replication cost between each pair of regions within $[0, 1]$, and a replication parameter $\beta = 10$. A region has a varying idle computation resource over time with CV randomly selected in $[0, 0.5]$, and the average amount of computation resource will be presented in the experiments.

Baseline Algorithm. We compare our design with a general pre-transcoding and load-based redirection strategy: (1) For segment transcoding, all versions of the videos are transcoded before publication, and each transcoded segment is replicated to 3 initial regions randomly selected (*i.e.*, the pre-transcoding scheme); (2) For user redirection, when requesting a segment, a user is redirected to a region which currently has the highest available upload bandwidth (*i.e.*, the load-based redirection scheme).

5.2 Experiment Results

5.2.1 Saving of Computation Resource

In this experiment, we assume that the CDN can provide unlimited computation resource when transcoding is performed, such that we can satisfy all the segment requests of users. In Fig. 14, the curves represent computation resource saved by our design under different number of versions, compared with the pre-transcoding scheme. In particular, each sample is the fraction of computation resource that has been saved over the computation resource required to transcode videos to all the versions till a simulation round. We observe that as the number of versions increases, the computation resource saved by online transcoding increases, *e.g.*, over 90% of the computation resource can be saved when the number of versions is over 8. The reason is that transcoding segments with no viewer to many versions costs a large amount of computation resource. We also observe that the amount of computation resource saving decreases in the first several rounds when new videos are published, and becomes stable afterwards. The reason is that in our design, computation resource is mainly used to transcode the most popular segments after the videos are published, and users who watch videos later largely request the segments that have already been transcoded.

Then, we investigate the impact of the number of videos published each time and the number of segments in each video. In this experiment, we fix the number of versions to 4. In Fig. 15, each bar represents the computation resource saving when a particular number of videos are published each time. We observe that publishing a large number of videos per time slot generally leads to larger computation resource saving. The reason is that the popularity distribution of the videos is heavy-tailed, and more videos with no viewer cause more waste of computation resource with the pre-transcoding scheme.

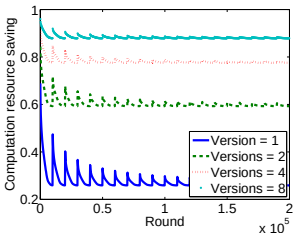


Fig. 14. Computation resource saved by online transcoding under different number of versions.

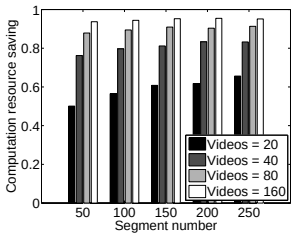


Fig. 15. Computation resource saved by online transcoding under different numbers of videos and segments.

5.2.2 Improvement of Quality of Experience

Taking advantage of online transcoding, users are able to be redirected to their ideal regions to achieve improved download speeds, and to receive segments with bitrates that best fit their download speeds. Next, we evaluate the improvement of the overall quality of experience.

Online transcoding allows users to be statically redirected to servers that serve particular segments, and improves the download speeds of users. We compare our redirection strategy with the load-based redirection scheme. In Fig. 16, the curves plot the user download speeds achieved by different strategies and the difference between them, versus the rank of users. We observe that our strategy can effectively schedule users to their ideal regions, with an average 150 Kbps improvement of download bandwidth, as compared to the load-based redirection scheme. The reason is that the load-based redirection scheme only considers segment replication and available bandwidth of the regions, while our strategy allows users to choose their ideal regions.

With the improvement of download speeds, the startup delay for users to watch videos can be reduced. In this experiment, we assume users have to fill a buffer of 256KB before starting playing a video. In Fig. 17, we plot the CDFs of startup delays achieved by our design and the load-based redirection strategy. Our design reduces the startup delay by over 2.5 seconds on average against the load-based strategy. In particular, over 15% more users in the load-based strategy have to experience a startup delay larger than 10 seconds, than that in our design.

The quality of experience in adaptive video streaming also depends on bitrates of segments users receive, which determine the image quality of videos. We compare the best versions users receive under different redirection strategies. Again, we fix the number of versions to 4. As illustrated in Fig. 18, each curve represents the version downloaded versus the user rank. We observe that as many as 44.8% of the users receive a version of a higher bitrate with our strategy than that with the load-based redirection scheme. In particular, over 4.5x users receive the version with the highest possible bitrate with our redirection strategy than with the load-based redirection scheme.

5.2.3 Fitness of the Transcoded Segments

In the following experiment, we will evaluate the effectiveness of our transcoding task schedule, on how well the transcoded segments match the users' requests. We compare our transcoding scheduling scheme with an FIFO-based scheme, where the transcoding tasks are performed according to request arrivals in an FIFO manner. For a fair comparison, we assume that users have already been redirected to regions according to our redirection strategy for both schemes. By varying the average computation resource in the regions, we evaluate the fitness of the transcoded segments. In Fig. 19, each sample represents the average bitrate difference between the bitrates of the received version and the requested version at all users versus the average computation resource of the region, calculated as the average number of segments that can be generated by the region. Note that the real

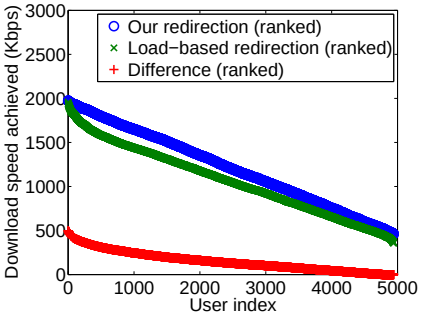


Fig. 16. Comparison of download speeds achieved at users under different redirection strategies.

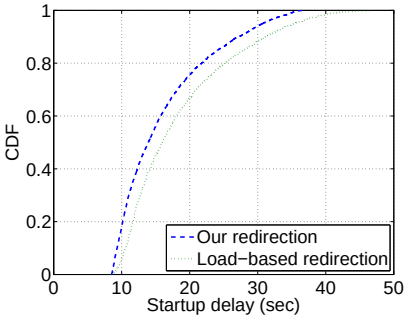


Fig. 17. Comparison of startup delays achieved by different redirection strategies.

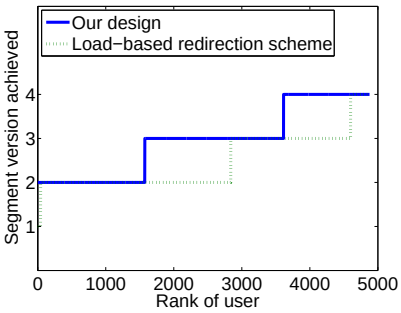


Fig. 18. Comparison of best versions received at users under different redirection strategies.

computation resource may be different across the regions as it takes different amount of computation resource to transcode different versions. A larger difference indicates a larger streaming quality degradation, as users have to receive a replacement segment with a much smaller bitrate. We observe that the average bitrate difference is much smaller with our design. In particular, our strategy can reduce the number of users who have to receive a segment of a mismatched version by over 42.2%.

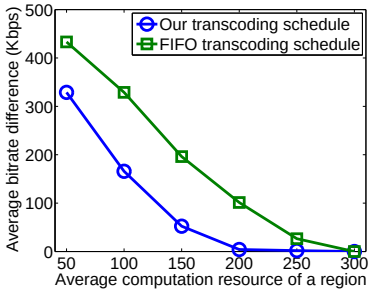


Fig. 19. Comparison of bitrate mismatch under different transcoding schedules.

To avoid users waiting for a segment that cannot be transcoded by the system on time, our design allows a low-version replacement segment to be sent to a user. We study the number of mismatch responses served to users since the publication of a set of 50 videos. The two curves in Fig. 20 represent the number of mismatched responses served to users over time. The number of mismatched versions for users with our design is 40% smaller, when the requested segments cannot be transcoded on time. Besides, the mismatched number of segments decreases over time as the transcoded segments are cached in our design.

5.2.4 Replication Cost

Our design utilizes regional preferences of versions when assigning transcoding tasks. Next, we evaluate the replication cost under different numbers of video versions. In Fig. 21, each curve represents the replication cost versus the number of versions, with a particular

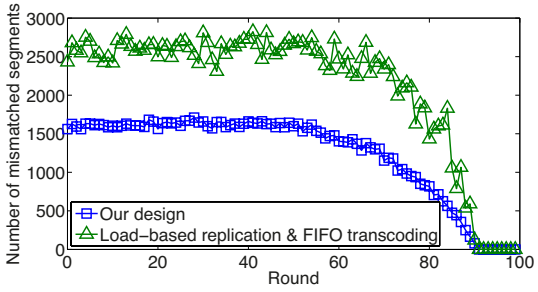


Fig. 20. Mismatch of versions of segments at users over time.

number of segments in each video. We observe that a larger number of versions leads to a smaller replication cost. The reason is that when more versions are available, our design can effectively allow regions to transcode heterogeneous versions that best meet their users' demand. We also observe that the number of segments has little impact on the replication cost, implying that we can use a small amount of time for adaptive scheduling without incurring increased replication cost. As more and more versions are used in today's adaptive streaming systems, our design reduces not only the waste of computation resource for transcoding, but also the replication cost of the transcoded segments.

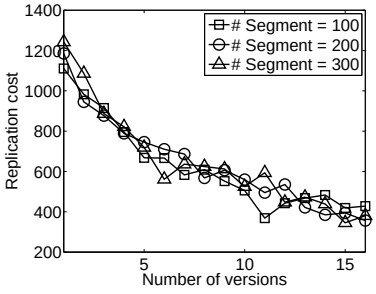


Fig. 21. Average replication cost per time slot versus the number of versions.

5.2.5 Performance Difference Between the Optimal Solution and our Algorithm

Our heuristic user-redirection algorithm is based on the strategy that users greedily choose the best regions that can serve them in a distributed manner, and servers also serve users in a best-effort way. We study how this strategy will perform compared with the optimal solution. As illustrated in Fig. 22, each bar represents the performance difference, i.e., the difference of $\sum_{u\in\mathbf{U}^{(T)},r\in\mathbf{R}} H_{u,r}D_{u,r}^{(T)}$ achieved by our design, and the optimal solution based on a brute-force search. The average difference is 15.2%, indicating that our distributed algorithm can find most of the users their best regions to receive the segments.

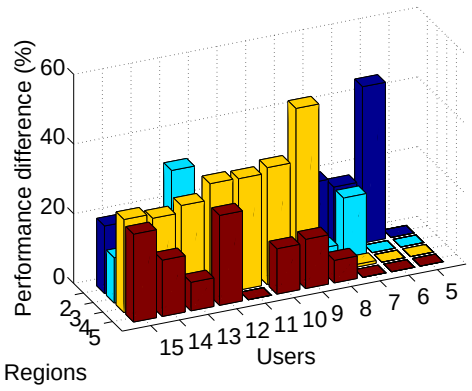


Fig. 22. Redirection performance difference between optimal solution and our algorithm.

6 Concluding Remarks

The rapid move to HTTP-based adaptive streaming requires dedicated strategies for improving the streaming quality for users and reducing the operation costs for content providers. Transcoding and delivery have been separately studied for adaptive video streaming, resulting in a significant waste of computation resource to transcode useless segments, and suboptimal streaming quality due to homogeneous replication of segments of different versions. Motivated by extensive measurement studies on both professional and social video services, we propose a joint online transcoding and geo-distributed delivery strategy, which allows us to explore a new design space for adaptive video streaming. We connect video transcoding and video delivery based on users' preferences of CDN regions and regional preference of versions to transcode. Aware of users' preferences of CDN regions, our design strategically performs user redirection so that videos can be streamed at large bitrates to the users. Taking into consideration heterogeneous importance of segments and regional preferences of versions to transcode, our design carefully schedules the transcoding tasks so that segments are transcoded to satisfy users' demands in each region, with little need of cross-region replication. Optimization problems are formulated and their hardness has been analyzed; we design heuristic and distributed algorithms to solve them. Our trace-driven experiments demonstrate significantly lowered computation resource consumption for segment transcoding, improved streaming quality for users, and reduced replication cost for video delivery, all with our design.

Acknowledgment

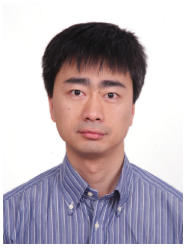
This work is supported in part by the National Basic Research Program of China under Grant No. 2015CB352300, the National Natural Science Foundation of China under Grant No. 61402247, 61472204 and 61210008, SZSTI under Grant No. JCYJ20140417115840259, Hong Kong RGC (HKU 717812E), and the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. We also thank BesTV for providing the valuable traces used in this paper.

References

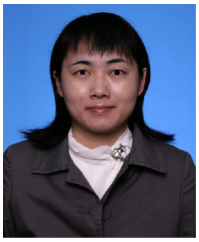
[1] ISO/IEC 23009-1:2014, "Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," 2012.
[2] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *ACM International Conference on Multimedia Systems Conference (MMSys)*, 2011.
[3] Z. Li, Y. Huang, G. Liu, F. Wang, Z. Zhang, and Y. Dai, "Cloud Transcoder: Bridging the Format and Resolution Gap between Internet Videos and Mobile Devices," in *ACM Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 2012, pp. 33–38.
[4] F. Lao, X. Zhang, and Z. Guo, "Parallelizing Video Transcoding Using Map-Reduce-Based Cloud Computing," in *IEEE International Symposium on Circuits and Systems*, 2012.
[5] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1357–1370, 2009.
[6] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2012.
[7] Bestv, "http://www.bestv.com.cn/."
[8] "http://weishi.qq.com/."
[9] Tencent, "http://www.tencent.com/."
[10] Z. Wang, L. Sun, C. Wu, W. Zhu, and S. Yang, "Joint Online Transcoding and Geo-distributed Delivery for Dynamic Adaptive Streaming," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2014.
[11] G. Peng, "CDN: Content Distribution Network," *arXiv preprint cs/0411069*, 2004.
[12] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid: a Framework for Video Transmission and Quality Evaluation," in *Computer Performance Evaluation. Modelling Techniques and Tools*. Springer, 2003, pp. 255–272.
[13] Z. Wang, A. C. Bovik, and L. Lu, "Why is Image Quality Assessment so Difficult?" in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.
[14] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "QDASH: a QoE-aware DASH system," in *ACM International Conference on Multimedia Systems Conference (MMSys)*, 2012, pp. 11–22.
[15] A. Vakali and G. Pallis, "Content Delivery Networks: Status and Trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.
[16] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49, no. 1, pp. 101–106, 2006.
[17] N. Wu, M. Wen, W. Wu, J. Ren, H. Su, C. Xun, and C. Zhang, "Streaming HD H.264 Encoder on Programmable Processors," in *ACM International Conference on Multimedia (Multimedia)*, 2009.

[18] Z. Huang, C. Mei, L. Li, and T. Woo, "CloudStream: Delivering High-Quality Streaming Videos Through a Cloud-Based SVC Proxy," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2011.
[19] J. Cohen, T. Repantis, S. McDermott, S. Smith, and J. Wein, "Keeping track of 70,000+ servers: the akamai query system," in *International conference on Large installation system administration*. USENIX Association, 2010, pp. 1–13.
[20] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. P. Markatos, and C. Dovrolis, "Available Bandwidth Measurement As Simple As Running wget," in *Passive and Active Measurement Conference (PAM)*, 2006.
[21] B. Cheng, H. Jin, and X. Liao, "Supporting VCR functions in P2P VoD services using ring-assisted overlays," in *IEEE ICC*, 2007.
[22] G. P. Zhang, "Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.
[23] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *ACM SIGCOMM*, 2008.
[24] C. Zheng, G. Shen, and S. Li, "Distributed Prefetching Scheme for Random Seek Support in Peer-to-Peer Streaming Applications," in *ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming*, 2005.
[25] L. Liang, "The Cloud Video Material Transfer Code System Design in the Global Station Network Environment," in *IEEE International Conference on Image Analysis and Signal Processing (IASP)*, 2012, pp. 1–3.

Zhi Wang (S'10 M'14) received his B.E. and Ph.D. degrees in Computer Science in 2008 and 2014, from Tsinghua University, Beijing, China. He is currently an assistant professor in Graduate School at Shenzhen, Tsinghua University. His research areas include online social media, mobile cloud computing and large-scale multimedia systems. He received Outstanding Doctoral Dissertation Award from China Computer Federation, Best Paper Award at ACM Multimedia 2012, and Best Student Paper Award at MMM 2015.



Lifeng Sun received his B.S. and Ph.D. degrees in System Engineering in 1995 and 2000 from National University of Defense Technology, Changsha, Hunan, China. He was an postdoctoral fellow from 2001 to 2003, an assistant professor from 2003 to 2007, an associate professor from 2007 to 2013, and currently a professor all in the Department of Computer Science and Technology at Tsinghua University. His research interests lie in the areas of online social network, video streaming, interactive multi-view video, and distributed video coding. He is a member of IEEE and ACM.



Chuan Wu received her B.E. and M.E. degrees in 2000 and 2002 from Department of Computer Science and Technology, Tsinghua University, China, and her Ph.D. degree in 2008 from the Department of Electrical and Computer Engineering, University of Toronto, Canada. She is currently an associate professor in the Department of Computer Science, the University of Hong Kong, China. Her research interests include cloud computing, peer-to-peer networks and online/mobile social network. She is a member of IEEE and ACM.

Wenwu Zhu (M'97 SM'01 F'10) received the Ph.D. degree from New York University Polytechnic School of Engineering in 1996 in Electrical and Computer Engineering. He is with Computer Science Department of Tsinghua University as Professor of "1000 People Plan" of China. His current research interests are in the area of multimedia cloud computing, social media computing, multimedia big data, and multimedia communications and networking. He served(s) on various editorial boards, such as Guest Editor for the Proceedings of the IEEE, IEEE T-CSVT, and IEEE JSAC; Associate Editor for IEEE Transactions on Mobile Computing, IEEE Transactions on Multimedia, and IEEE Transactions on Circuits and Systems for Video Technology; Leading Editor of the Area "Computer Networks and Distributed Computing" of Journal of Computer Science and Technology. He received the Best Paper Award in ACM Multimedia 2012, the Best Paper Award in IEEE Transactions on Circuits and Systems for Video Technology in 2001, and the other 4 international Best Paper Awards.

Qidong Zhuang Qidong Zhuang received his B.Eng. degree in computer science from Shenyang Institute of Engineering, Shenyang, Liaoning, China, in 2012. He worked as an engineering, a researcher and a system architect at different departments of Tencent from 2012 to 2014. He is currently pursuing the M.Phil. degree in Technology Leadership and Entrepreneurship at the School of Engineering, The Hong Kong University of Science and Technology.

Shiqiang Yang received the B.E. and M.E. degrees in Computer Science from Tsinghua University, Beijing, China in 1977 and 1983, respectively. From 1980 to 1992, he worked as an assistant professor at Tsinghua University. He served as the associate professor from 1994 to 1999 and then as the professor since 1999. From 1994 to 2011, he worked as the associate header of the Department of Computer Science and Technology at Tsinghua University. His research interests mainly include multimedia procession, media streaming and online social network. He is a senior member of IEEE.