Computer Networks 75 (2014) 86–98

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Distributed joint optimization for large-scale video-on-demand ☆

Dongni Ren [a,*], S.-H. Gary Chan [a], Guangyu Shi [b], Hongbo Zhang [b]

[a] The Hong Kong University of Science and Technology, Kowloon, Hong Kong
[b] Huawei Technologies Co., Ltd, Shenzhen 518129, China

ARTICLE INFO

Article history:
Received 12 October 2013
Received in revised form 30 August 2014
Accepted 29 September 2014
Available online 12 October 2014

Keywords:
Video on demand
Content replication
Storage planning
Server selection

ABSTRACT

We study the provisioning of large-scale video-on-demand (VoD) services to distributed users. In order to achieve scalability in user capacity overcoming the limitation in core network bandwidth, servers are deployed close to user pools. They replicate movie segments cooperatively under the constraint of their storages. Considering the realistic scenario that access delay is a function of the total traffic in the underlay link (including cross-traffic), we address the following optimization issues in the server overlay: (1) Which segments should each server replicate to achieve network-wide good locality effect? This is the so-called *content replication (CR)* problem; (2) Given a segment miss at a server and a number of remote servers storing the segment, which of them should serve the local server to conserve network bandwidth? This is the so-called *server selection (SS)* problem; and (3) Given a certain total storage budget in the VoD network, what should be the capacity allocated to each server to achieve low access delay? This is so-called *storage planning (SP)* problem. Clearly the decisions of CR, SS and SP are inter-dependent, and hence need to be *jointly* optimized.

We first formulate the joint optimization problem and prove that it is NP-hard. We then propose a simple and distributed algorithm called CR–SS–SP to address it. CR–SS–SP achieves good storage allocation, replicates segments collaboratively and adaptively to achieve high locality, and selects servers efficiently with a simple lookup. Simulation results on both Internet-like and real ISP topologies show that CR–SS–SP significantly outperforms existing and state-of-the-art approaches by a wide margin (often by multiple times).

1. Introduction

With the penetration of residential broadband network, providing video-on-demand (VoD) service to distributed users has attracted much attention recently [1–4]. Numerous on-demand Internet movie applications have been deployed, and most of them make use of data centers or Content Delivery Networks (CDN) for content distribution to their end users [5]. We consider a content or service provider offering large-scale VoD service. There are several challenges in deploying such video distribution network, e.g., how to efficiently replicate and access contents so as to achieve the best user experience. In this paper, we consider a scalable server architecture for large-scale VoD, and study its distributed joint optimization issues. As we consider a managed VoD network deployed by a service or content provider (with bandwidth allocated between servers), addressing issues regarding link instability and DNS are less critical and outside the scope of this work.

☆ This work was supported, in part, by Hong Kong Research Grant Council (RGC) General Research Fund (610713) and The Hong Kong Innovation and Technology Fund (UIM/246).

* Corresponding author.
E-mail addresses: tonyren@cse.ust.hk (D. Ren), gchan@cse.ust.hk (S.-H.G. Chan), shiguangyu@huawei.com (G. Shi), zhanghongbo888@huawei.com (H. Zhang).

http://dx.doi.org/10.1016/j.comnet.2014.09.014
1389-1286/

We show in Fig. 1 the VoD network under consideration. There is a repository storing all the video contents. To scale up the streaming or user capacity, distributed servers are deployed close to user pools forming an overlay network. Such deployment also substantially reduces the load of the core network due to the decrease in the long-haul connections between users and servers. In the network, the clients may be set-top boxes, Internet TVs or PCs connected to the servers (through, for example, some Digital Subscriber Line Access Multiplexer (DSLAM)). To facilitate storage, the movies are divided into fixed-sized *segments*. Users may at any time perform random seeks to any movie segments using interactive DVR functionalities. Each user has a *home server*, which is his only contact point to the VoD service. The home servers stream to the user directly if they have replicated locally the segment of interest (we consider the usual case that the bandwidth between the users and their home servers is not a bottleneck). Otherwise, the home server re-directs the request to another server (including repository) storing the segment, which is called the *remote server*. The remote server then streams directly to the user once network bandwidth is available. Such remote server is the *streaming parent* of the home server. For example the rightmost server in the figure is the home server of the clients that connect to it. For those clients, all other servers in the network are remote servers.

Note that each movie segment may have different access probabilities which may slowly vary over time. (Though the segment may be 5–30 min of movie, our study is by no means limited to that. During movie streaming, the video segments can be further "packetized" into smaller chunks for efficient transmission using adaptive http live streaming (HLS or DASH)). Due to the content volume and its skewed popularity, it is not cost-effective, and often not possible, to replicate all the segments in all the servers. For example, the storage of a 100-min 5 Mbps movie is about 4 GB. Therefore, to store 10,000 movies, a server requires a capacity of 40 TB, which is not cost-expensive to maintain locally, especially in a dynamic environment when most of the movies are not very popular. The server overlay cooperates in a "peer-to-peer" manner by streaming segments to fulfill each other's local requests. In this paper, we use "client," "request," and "user" interchangeably.

We consider the realistic scenario that the delay of an underlay link is a function of its total traffic including the cross traffic. There are three important issues which need to be addressed. First, which segments should a server replicate given its storage to achieve good segment locality? This is the so-called *content replication (CR)* problem. Second, in case of a miss in the home server and given a number of available remote servers storing the requested segment, which of them should the home server choose to achieve good overall user delay experience? This is the so-called *server-selection (SS)* problem. Third, given a certain total storage budget in the system, what storage capacity should be allocated to each server? This is the so-called *storage planning (SP)* problem.

Clearly, the decisions of CR, SS and SP are interdependent, i.e., the optimal strategy of one depends on the

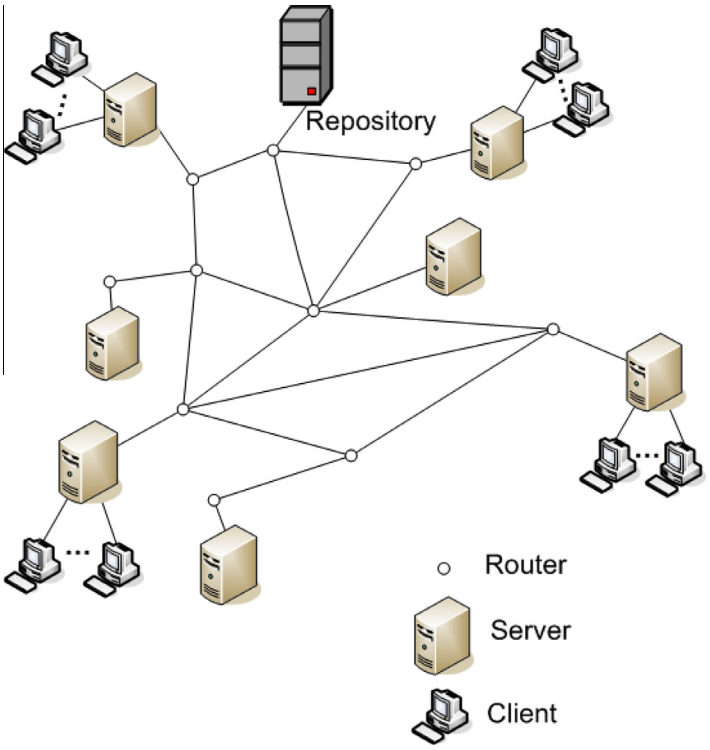

Fig. 1. A distributed server approach for VoD service.

strategies of the other two. A joint optimization of CR, SS and SP is hence necessary to achieve the best overall performance. In this work, we address this joint optimization problem with the following contributions:

- *Problem formulation and its complexity analysis:* Using a realistic overlay network model, we formulate the problem of jointly optimizing CR, SS and SP to minimize user delay. We prove that this problem is NP-hard.
- *CR–SS–SP, a distributed joint optimization algorithm:* Note that, given a total storage budget, there exists many combinations of server storage, each of which leads to an optimal user delay. In order to efficiently identify a solution to minimize user delay, we propose a distributed heuristic called CR–SS–SP to address the joint optimization problem. Note that CR–SS–SP works even if SS, CR and SP are in different time scales. For example, consider the usual case that they are in increasing order of magnitude. First CR–SS–SP is jointly optimized with the time scale of SP, then CR–SS can be optimized with the time scale of CR, and then finally SS is optimized in the shortest time scale given SP and CR.
  CR–SS–SP is based on only simple overlay measurement (i.e., without the knowledge of underlay topology) without the need of full network information. Servers may run the algorithm in parallel and independently to come up with good CR, SS and SP strategies. It is fully distributed (i.e., scalable to large network) and simple to implement. By exchanging with its neighbors a routing table of its stored segments, a server can efficiently find good neighbor(s) for any of the segments and make efficient replication decisions. Besides CR and SS strategies, CR–SS–SP also jointly designs server storage given a certain network-wide total storage budget. It is a truly joint algorithm where given one, it optimizes the other two. CR–SS–SP is guaranteed to converge, and has low server and computational overheads.
- *Performance study of CR–SS–SP:* We conduct extensive simulation study of CR–SS–SP on both Internet-like and real network topologies. Our results show that CR–SS–SP outperforms both traditional and state-of-the-art schemes by a wide margin, achieving substantially lower user delay and bandwidth usage (often cut by more than 70% better).

This paper is organized as follows. In Section 2 we review related work. In Section 3 we formulate the joint optimization of CR, SS and SP, and prove that it is NP-hard. We present the distributed CR–SS–SP algorithm and its exchange overhead and computational complexity in Section 4. In Section 5 we present illustrative simulation results on the performance comparison of CR–SS–SP. We conclude in Section 6.

2. Related work

There has been much work on VoD based on efficient file downloading [6–8]. In contrast, we consider interactive *streaming* with content replication, server selection and storage planning. Other work on VoD focuses on the design of its infrastructure and framework [9–17]. Although this body of work contains some of SP, CR and SS, it has not addressed their *joint optimization*. In contrast, our work presents mathematical formulation and distributed joint optimization of SP, CR and SS.

The SS problem has often been studied in the context of overlay routing problem in the literature [18–22]. This body of work solves SS based on overlay measurement (e.g., PING and traceroute) or using information from ISP without considering the impact of CR, an important problem which we consider here. There has been work to optimize independently CR [23–26] and SS [27]. However, they have not considered their joint optimization with SP.

Traffic engineering has been studied together with cooperative distribution [28,29]. These work studies content distribution from a ISP's point of view by coordinating traffic engineering (to select efficient routes for the traffic) and server selection (to match servers with subscribers). These works assume *fully replicated* contents in each server. Our work, on the other hand, is to consider partial replication in the servers, which leads to joint optimization of content replication and server selection. We solve the problem as content providers, assuming given routes in the underlay.

There have been extensive studies on data replication and server selection in CDNs [30]. The work in [31] formulates the constraints such as bandwidth and storage into a 5 dimensional search space, and use a Particle Swarm Optimization algorithm to find the most appropriate server to store each replica. The work in [32] models the server selection problem as a sequential decision problem solved by the Multi-Armed Bandit paradigm. However, most of the works consider content replication and server selection as separate problems, and solve them independently. Furthermore, storage planning (SP) has not been considered with CR and SS as well. In this work we jointly optimize the three operations in the VoD network together, and propose efficient algorithms for each operation.

Joint optimization of CR and SS has been discussed in [33–37], but none of them has considered SP, which is an important problem in VoD. How content providers allocate the storage budget to different servers directly determines the system performance such as replicate miss rate and user delay. Besides given a storage planning solution, it also affects the decisions for content replication and server selection. The work [33,34] assumes that overlay end-to-end delay is not affected by the underlay link traffic. We differ by considering the more realistic scenario where the delay of the underlay links (including propagation and queuing delay) may be a non-linear function of the link traffic. As a result, the underlay cross traffic has an impact on the overlay end-to-end delay. This calls for a totally different problem formulation and algorithmic approach. The work of [35,36] assumes the server network to be a specific tree topology with homogeneous link bandwidth and user demand. It cannot be extended to a general mesh network topology with heterogeneous link bandwidth and user demand as we consider here. Furthermore, it has not taken into account the influence of cross traffic in the underlay which we consider here. Some other work is centralized in nature [38], which relies on complete network information and hence does not scale well to large (and dynamic)

networks. We present a distributed algorithm for the server overlay where the servers cooperate in a peer-to-peer manner to improve access delay. Our algorithm is scalable as each server only needs partial information (i.e., user delay and routing table) from its neighbors. As compared to [37] which studies the joint optimization of CR and SS, we present here an effective distributed solution for CR, SS and SP. We also present a mathematical formulation for the joint optimization problem, which is proved to be NP-hard.

3. Problem formulation

We focus on the server network and model the underlay network as a graph $G = (V, E)$, where V is the set of nodes including servers and underlay routers. E is the set of directed underlay links between them. The repository is considered as a regular server that stores all the video segments. In the case of a miss in the home server, if no remote server replicates the requested segment, the request goes to the repository. We show the important symbols that are used in this paper in Table 1.

Let S be the set of servers (including the repository) and R be the set of routers. Obviously, we have

$$V = S \cup R, \tag{1}$$

and

$$S \cap R = \emptyset. \tag{2}$$

We consider video segments are of the same size so that storage, scheduling and transmission can be efficiently planned and organized. Let T be the set of video segments, and C be total storage budget. A server i has a certain storage of c_i segments, where $i \in S,\ c_i \in Z^+$ and $c_i \leqslant |T|$. Let $I_i^j \in \{0, 1\}$ be a binary variable indicating whether server i replicates segment j. We obviously must have

$$\sum_{i \in S} c_i \leqslant C, \tag{3}$$

and

$$\sum_{j \in T} I_i^j \leqslant c_i, \quad \forall i \in S. \tag{4}$$

Each user is associated with a home server. We assume the user requests are independent. Let λ_i^j be the popularity of segment j at server i, which is defined as the probability that a user request accessing segment j at server i. Clearly, we must have

$$\sum_{j \in T} \lambda_i^j = 1, \quad \forall i \in S. \tag{5}$$

Let D_i be the total streaming bitrate serving all the users homed at server i (bits/s), which means that server i is this group of users' home server. Therefore, the average total bitrate for segment j at server i is given by $\lambda_i^j D_i$.

Consider server i with a missed request of segment j, and a remote server k has the segment j. We consider the general case that multiple servers may supply simultaneously the stream to the request. Let r_{ik}^j be the fraction of this bitrate supplied by server k, where $0 \leqslant r_{ik}^j \leqslant 1$. In order to aggregate a stream, we must have

$$\sum_{k \in S} r_{ik}^j I_k^j = 1, \quad \forall i \in S,\ j \in T. \tag{6}$$

When $I_k^j = 0,\ r_{ik}^j$ needs to be 0 as well. In order to ensure that those servers *without* segment j do not contribute to the bitrate, we further need

$$\sum_{k \in S} r_{ik}^j = 1, \quad \forall i \in S,\ j \in T. \tag{7}$$

Denote the total amount of VoD traffic from server k to serve the users homed at server i as t_{ki}, we have

$$t_{ki} = \sum_{j \in T} r_{ik}^j \lambda_i^j D_i, \quad \forall i, k \in S. \tag{8}$$

Furthermore, for traffic/flow conservation, we need

$$\sum_{i,k \in S} t_{ki} = \sum_{i \in S} D_i. \tag{9}$$

The routing strategy in the underlay network can be represented by $\left\{f_{ki}^l\right\}$, where f_{ki}^l is the fraction of the traffic from k to i that goes through link l ($0 \leqslant f_{ki}^l \leqslant 1$). Given $\left\{f_{ik}^l\right\}$, the traffic on link l brought by t_{ki} is then given by $t_{ki} f_{ki}^l$. The aggregated traffic on link l is therefore

$$F_l = \sum_{i,k \in S} t_{ki} f_{ki}^l, \ \forall l \in E. \tag{10}$$

Denote the propagation delay and queuing delay on link l by d_l^p and d_l^q, respectively [39]. Then we have

$$d_l = d_l^p + d_l^q, \quad \forall l \in E. \tag{11}$$

Table 1
Major symbols used in this paper.

Notation	Definition
λ_i^j	Access probability of segment j at server i
c_i	Storage capacity of server i (in number of segments)
C	Total storage budget in the network (in number of segments)
I_i^j	0 or 1 variable indicating if server i replicates segment j
D_i	Total streaming bitrate serving the users homed at server i (bits/s)
r_{ik}^j	The proportion of requests for segment j served by server k to users homed at server i
f_{ik}^l	The proportion of the traffic from server i to server k that passes underlay link l
d_l^p	Propagation delay of link l (s)
d_l^q	Queuing delay of link l (s)
d_l	$d_l = d_l^p + d_l^q$, the total delay of link l (s)
σ	Packet size in the network (bits)
b_l	Bandwidth of link l (bits/s)
d_w	Total delay of path w (s)
P_{ik}^j	The probability that server k is chosen as the streaming parent of server i for segment j
$\bar{d}_i^j$	The average access delay for segment j at server i (s)
$\bar{d}_i$	The average access delay of segments at server i (s)
$\bar{d}$	The average access delay of all segments in the network (s)
$\bar{d}_{L_i}$	The average segment access delay in the local region L_i formed by the neighbors of server i (s)
G_i^j	Sum of average segment j's delay change in the server i's local region L_i (s)
t_{ki}	The total amount of VoD traffic from server k to serve the users homed at server i (bits/s)

Let d_w be the delay of the path w. Obviously, we have

$$d_w = \sum_{l \in w} d_l. \tag{12}$$

Let H_{ki}^w be the fraction of t_{ki} that goes through w. Note that user delay at a home server depends on the path delay between the segments and the server, the so-called *segment delay*. We hence minimize the average segment delay $\bar{d}$, which can be written as the sum of path delay weighted by the proportion of traffic:

$$\bar{d} = \frac{\sum_{i,k \in S} t_{ki} \sum_w H_{ki}^w d_w}{\sum_{i,k \in S} t_{ki}}. \tag{13}$$

Using Eqs. (9) and (12), we can rewrite the numerator as a sum over each link l, i.e.,

$$\bar{d} = \frac{\sum_{l \in E} F_l d_l}{\sum_{i \in S} D_i}. \tag{14}$$

We state the joint optimization problem as follows:

Joint CR–SS–SP Problem: Given user demand $\{D_i\}$, total storage budget $\{C\}$, segment popularity $\{\lambda_{ij}\}$, underlay routing $\left\{f_{ik}^l\right\}$ and topology G, we seek to minimize $\bar{d}$ given in Eq. (14), subject to Eqs. (3), (4), (6)–(8), (10) and (11). The output is the SP strategy $\{c_i\}$, CR strategy $\left\{I_i^j\right\}$ and SS strategy $\left\{r_{ik}^j\right\}$.

Claim 1. *The Joint CR–SS–SP Problem is NP-hard.*

Proof. We prove its NP-hardness by deriving a polynomial reduction from the Domatic Number Problem, which is NP-complete [40]. Its NP-complete version is stated as follows. Given $G = (V, E)$ and a positive integer K no larger than $|V|$, is the domatic number of G at least K, i.e., can V be partitioned into at least K disjoint dominating sets? Given G and K, we construct an instance of our decision problem as follows. The network contains K video segments with equal popularity everywhere, with the total storage budget $C = |V|$. Consider the case that each link has infinite bandwidth (i.e., queuing delay is zero) and constant propagation delay of 1 unit. The instance construction can obviously be done in polynomial time. The average access delay is larger than or equals to $(K-1)/K$, and the minimum delay can be achieved when each server replicates exactly one segment. Our decision problem becomes: Given the instance, is there a joint CR–SS–SP strategy which achieves average access delay of at most $(K-1)/K$? Using"proof-by-restriction" method [41,42], we only need to show Domatic Number Problem is reducible to this special instance. □

At each node v, the minimum average access delay is $(K-1)/K$ when one segment is stored locally and $(K-1)$ segments are one hop away. We show that G can be partitioned into K disjoint dominating sets if and only if there is such a strategy. First, if there is such a strategy, any node v can access a segment within one hop distance. Then if we separate V into K disjoint sets with each corresponding to a certain video segment, we must result in K dominating sets because the distance between a set and any node is either zero (i.e., contained in the set) or one unit (i.e., connected by an edge). Furthermore, if we have K disjoint dominating sets, we can easily derive such a strategy by assigning each set a distinct segment to replicate. Therefore, we reduce the Domatic Number Problem to a special case of our decision problem, and it proves that our optimization problem is NP-hard.

4. CR–SS–SP: distributed joint optimization algorithm

In this section, we present our distributed heuristic called CR–SS–SP to address the NP-hard problem. In CR–SS–SP, each server independently minimizes the average segment delay in its local region through segment replacement. The joint optimization is based on three algorithms built successively as follows: first SS given CR and SP, then CR–SS given SP, and finally CR–SS–SP together. CR, SS and SP are jointly optimized by minimizing a delay function iteratively in a distributed manner, and we prove that CR–SS–SP is guaranteed to converge. In Sections 4.1–4.3, we present our probabilistic server selection framework, content replication and storage planning used in CR–SS–SP, respectively. In Section 4.4, we discuss the exchange overhead and computational complexity of CR–SS–SP.

4.1. Probabilistic SS framework

This framework optimizes SS given CR and SP. Every server has a simple routing table, each entry of which corresponds to a video segment it stores. For each video segment, the row contains some known servers with the segment and hence can be chosen as its streaming parent. In order to reduce the size of the table, we keep a maximum number n_s of such known servers for each segment. Each server periodically exchanges its routing table with its neighbors, and pings the path delay to them. Let E_i^j be the set of servers stored in the entry for segment j at server i. Let d_{ik}^j be the access delay of segment j from server k for users homed at server i. For each server $k \in E_i^j$, apart from its IP address, the entry also stores a utility value indicating the importance of server k as U_{ik}^j. Obviously, U_{ik}^j should be a monotonically decreasing function in d_{ik}^j. For concreteness in our study, we choose

$$U_{ik}^j = \begin{cases} 1/d_{ik}^j, & \text{if } j \text{ is stored at } k; \\ 0, & \text{otherwise;} \end{cases} \tag{15}$$

and

$$d_{ik}^j = d(i,k), \tag{16}$$

where $d(i,k)$ is the path delay from server k to server i, which can be inferred by a "ping" operation between i and k (by definition, $d(i,i) = 0$). Note that our algorithms are not restricted to this particular form of the function; any other function of U_{ik}^j may be used. In order to choose close servers as streaming parents and not to direct all the traffic to a particular server, we adopt a probabilistic framework to achieve load spreading. In the framework,

the probability that server k is chosen as the streaming parent for segment j is given by

$$P_{ik}^{j} = \frac{U_{ik}^{j}}{\sum_{t \in E_i^j} U_{it}^{j}}. \quad (17)$$

Clearly from the equation, if the home server has replicated the segment, it has probability 1 to become the streaming parent (and hence surely serves its local users). On the other hand, if the segment is not found in the home server, only the servers with the segment are considered to become the parent: the larger the path delay, the lower is a server's probability to become the streaming parent.

4.2. Joint CR–SS given SP

Given server storage (as in SP), CR–SS–SP jointly optimizes CR and SS. The SS strategy is based on the information in the routing table. Consider a user request sent to its home server i for segment j. Server i then inspects the parent list in the corresponding entry of its routing table. Each server stored in that list has a probability to be chosen as streaming parent as given by Eq. (17).

From Eq. (17), the expected segment delay for users at home server i to get segment j is given by

$$\bar{d}_{ij} = \sum_{k \in E_i^j} P_{ik}^{j} d_{ik}^{j}. \quad (18)$$

Given the segment popularity $\left\{\lambda_i^j\right\}$, the average segment access delay at server i is hence given by

$$\bar{d}_i = \sum_j \lambda_i^j \bar{d}_i^j. \quad (19)$$

Each server has a certain number n of neighbors, which are the closest servers according to the delay. Denote this set of neighbors for server i as N_i. The local region L_i is defined as the set including server i and its neighbors in N_i. From Eq. (14), the average access delay in L_i is hence given by

$$\bar{d}_{L_i} = \frac{\sum_{k \in L_i} D_k \bar{d}_k}{\sum_{k \in L_i} D_k}. \quad (20)$$

In CR–SS–SP, each server i independently minimizes the average segment access delay in its local region (i.e., $\bar{d}_{L_i}$) through content replacement presented below. As a result, the servers jointly minimize the average segment delay $\bar{d}$.

The content replication strategy of CR–SS–SP includes two parts, i.e., segment initialization and segment replacement. For each server, given the storage allocated in Section 4.3, the segment initialization assigns video segments at server startup. The assignment can be either random or based on segment popularity. In any case, CR–SS–SP converges to a steady state (that each server replicates the appropriate segments) independent of the initial state. For concreteness in our simulation study, we randomly assign video segments to each server at startup.

Each server asks its neighbors to compute the delay change for each segment, which is, for the users homed at the neighbor server, the change in access delay due to replication or replacing the segment. After getting the information of such change from its neighbors, for each segment it computes the gain which is sum of the delay change for each neighbor.

The principle is that, for a stored segment, a positive gain means replacing the segment will reduce the delay while negative one means that replacing the segment will increase the delay. On the other hand, for an unstored one, a positive gain means replicating the segment will reduce the delay while negative one means that replicating the segment will increase the delay.

We describe below the details of the above operation. Each server stores two lists:

- *Store List*, where each entry stores the ID of the segment that is stored in the server and corresponding gain.
- *Replacement List*, where each entry stores the ID of the segment that is not stored in the server and corresponding gain.

A segment in the Store List is replaced by another one in the Replacement List when the discrepancy of the gain is larger than a certain threshold.

The initial segment assignment is not likely to be optimal, and hence the server goes through segment replacement to improve the performance. A server decides which segment to replace using the routing tables exchanged from its neighbors. Given all its neighbors' routing tables, a server examines the gain (in terms of reduction in access delay) of a certain segment replacement.

Consider that server i checks whether to replace segment j by segment t. It examines the routing table obtained from its neighbor k. Recall that E_k^j is the entry which stores the parent list for segment j in server k. If $i \in E_k^j$, server i is possibly chosen by k as the streaming parent for segment j. However, if server i decides to replace j, server k has to remove i from E_k^j. The expected access delay of segment j at server k (i.e., $\bar{d}_k^j$ in Eq. (18)) also changes. Denote this access delay change of segment j at server k as C_{ik}^j.

Let $\bar{d}_k^{j^-}$ and $\bar{d}_k^{j^+}$ be the expected segment delay at server k before and after the replacement (i.e., without segment j and with j), respectively. The change of $\bar{d}_k$ due to removing server i from E_k^j is given by

$$C_{ik}^j = \lambda_k^j (\bar{d}_k^{j^-} - \bar{d}_k^{j^+}). \quad (21)$$

Similarly, i can be chosen as streaming parent for t after the replacement. The change of $\bar{d}_k$ due to adding server i into E_k^t is given by

$$C_{ik}^t = \lambda_k^t (\bar{d}_k^{t^+} - \bar{d}_k^{t^-}). \quad (22)$$

We further define the gain $G_i(j)$ as the sum of average segment j's delay change in server i's local region L_i. Given Eq. (20), we have

$$G_i(j) = \frac{\sum_{k \in L_i} D_k C_{ik}(j)}{\sum_{k \in L_i} D_k}. \quad (23)$$

Server i sorts the segments in the Store List and Replacement List according to their total gain. Server i examines the segment j with the lowest $G_i(j)$ in the Store List and the segment t with the highest gain $G_i(t)$ in the Replacement List. Because the number of stored segments is limited by the storage, the number of pairs that the server needs to examine is only $O(|T|)$.

We use a certain threshold value α to guarantee the convergence of our distributed algorithm. A replacement is applied only if $(G_i^t - G_i^j)/\sum_{j\in T} G_i^j$ is larger than α. If it is larger than α, server i discards the entry of segment j and the entry of segment t in the Store List and Replacement List, respectively, and continues to examine the next pair of segments. Otherwise, server i stops segment replacement. This threshold eliminates the tiny delta improvements of the gain. If α is set to zero, the replacement occurs whenever a segment has larger gain than the ones in the current Store List. In the worst case scenario there will be uncountable number of updates. However if we have this threshold parameter, each replacement increases the total gain by at least α (in proportion), therefore the algorithm is guaranteed to converge. The choice of alpha depends on the deployment requirements. It is a parameter that makes sensible tradeoffs between convergence time and optimality. A large value of alpha leads to less replication updates, and significant improvement on the convergence time. A small value of alpha will lead to a higher value of the total gain, and hence smaller average delay.

4.3. Joint CR–SS–SP algorithm

CR–SS–SP jointly optimizes CR, SS and SP in a distributed manner through an iterative approach which guarantees to converge. Initially the servers have certain storage (which can be the same) given a total storage budget. When a server, say server i, is to optimize its storage, it first asks its neighbors for the Store Lists. After receiving all the Store Lists, server i adds the IP address of the neighbor who sends the Store List to each entry, and combines all the entries in those Store Lists as a "Plan List". Then server i sorts the "Plan List" according to the gain of each entry. Let P_i be the Plan List of server i, and M_{ik}^j be the entry in P_i indicating that server k replicates segment j. Let J_i be the sum of the segment gain of each entry in P_i. We have

$$J_i = \sum_{(k,j)\in P_i} G_k^j. \tag{24}$$

Servers then trade storage to improve network performance. The details of CR–SS–SP is shown in Algorithm 1. Server i examines Let Segment j be the largest gain in P_i for server i, and segment j' be the lowest gain in Q_i for server k. If $(G_i^j - G_k^{j'})/J_i$ is larger than a certain threshold β ($\beta < 1$), server i increases its storage by 1 and replicates segment j. It then asks server k to discard segment j' by decreasing its storage by one. Server i repeats this process until all the segments in its Replacement List have been examined. We use α and β to guarantee the convergence for Store List calculation and storage adaptation with a time-optimality tradeoff.

Algorithm 1. CR–SS–SP algorithm

for server k in server i's local region L_i **do**
  for segment j in i's Store list **do**
    calculate the delay change C_{ik}^j;
  end for
  for segment t not in i's Replacement list **do**
    calculate the delay change C_{ik}^t;
  end for
  send its store list to server i
end for
for segment j in i's storage **do**
  calculate G_i^j;
  $total\_gain += G_i^j$;
end for
sort store list and replacement list according to G_i^j;
repeat
  select segment t' with lowest $G_i^{t'}$ in i's Store list
  select segment t with highest G_i^t in i's Replacement list
  if $\frac{G_i^t - G_i^{t'}}{total\_gain} \geqslant \alpha$ **then**
    replace segment t' by segment t at server i
  end if
until $\frac{G_i^t - G_i^{t'}}{total\_gain} < \alpha$
combines the store lists as "Plan List"
sort "Plan List"
calculate J_i
repeat
  select segment j with the highest G_i^j in i's Replacement List
  select segment j' stored in server k with highest $G_k^{j'}$ in i's Plan List
  if $\frac{G_i(j) - G_k^j}{J_i} \geqslant \beta$ **then**
    i increases its storage by 1, and replicates segment j'
  end if k decreases its storage by 1, and discards segment j
until $\frac{G_i(j) - G_k^j}{J_i} < \beta$

4.4. Exchange overhead and computational complexity

Note that CR–SS–SP has low control overhead because the servers do not need to exchange the entire routing table; they only need to exchange two types of information. First, when a server i performs segment replacement, it only needs the information of the delay change of a neighbor k given by Eqs. (21) and (22). The information of the potential delay change can be efficiently calculated at neighbors in $O(|T|)$ time and sent to the server with size $O(|T|)$. Second, after the replacement, the server just needs to send its segment replication information in a bitmap of $O(|T|)$ to each of its neighbors. Clearly, the exchange overhead of both types of information is rather low. The low exchange overhead may be illustrated with an example

below. For 1200 segments and the delay change for each segment is represented by a floating number of 4 bytes, the bitmap of segments is about 150 bytes and the total size of delay change is around 5 KB. If segment replacement is done every, say 10 min, and each server has 15 neighbors, the table exchange bandwidth is only around 1 kbits/s.

The computational load of our algorithm is also low. As mentioned above, each neighbor computes potential delay change in $O(|T|)$ time. After obtaining all such information, a server sorts them in $O(|T|\log|T|)$ time. For segment replacement, we simply calculate the reduction of delay by removing the segment of lowest gain and replacing it with the segment of highest gain. We repeat this process for the next entry until the reduction of delay is less than the threshold. The running time of the whole process is $O(|T|)$, because the segments have already been sorted. Hence the total running time of CR–SS–SP for replacement is $O(|T|\log|T|)$.

5. Simulation environment and illustrative results

5.1. Simulation environment and metrics

We have conducted extensive simulations to study the performance of CR–SS–SP on both Internet-like topology generated by BRITE (using the default Router Waxman model) and a real Internet topology obtained from an ISP [43]. The BRITE topology is a router-only topology, which contains 3072 routers and 10,850 underlay links. BRITE also gives us the link propagation in milli-seconds. Each underlay link has a certain link bandwidth. The real Internet topology is obtained from the Internet Initiative Japan (IIJ) backbone, the largest data network in Japan.

In our simulation, a certain number of servers are randomly attached to the routers in the topology. Each server has a certain number of nearest servers as its neighbors. All segments (or groups of movies) are of the same size and their popularities follow the Zipf distribution at each server, i.e., if $f(k;s,N)$ is the k-th highest of the segments, given the skewness parameter and T number of segments, then we have

$$f(k;s,T)=\frac{1/k^s}{\sum_{n=1}^{T}1/n^s}, \tag{25}$$

where s is the Zipf parameter. When s is small (e.g., $s=0$), all segments have similar popularities; when s is high (e.g., $s=1$), the distribution is more skewed. (Note that the use of Zipf distribution is for illustration only. Any other distribution including the long-tailed ones may be used.)

We write our event-driven simulation in Java. Segment requests arrive at a server according to a poisson process with rate λ (requests/minute). We consider that the delay in a link may be modeled as (Eq. (11))

$$d_l=d_l^p+\sigma/(b_l-F_l), \tag{26}$$

where σ is the average packet size (in bits) and b_l is the bandwidth of link l (in bits/s).

A server periodically replaces its stored segments. When a replacement decision is made, the server downloads the segment from another server according to the server selection algorithm. The downloading bitrate is higher than the streaming bitrate. While being downloaded, the segment may be played back. When downloading is completed, the server informs its neighbors its new routing table and the neighbors update theirs accordingly. A user may *renege* (i.e., leaves the system due to impatience) if a requested segment is not played back by a certain time.

For each experiment, we run it till steady state, after which we take the statistics. Unless otherwise stated, we use the default values according to Table 2 in our simulations. We will vary them to study their effects on system performance. Note that the performance scales for larger systems. For example, if we have 10 times as many segments, we may group 10 segments as one "super segment", and the resulting problem and approach are exactly the same as the current setting.

The performance metrics that we are interested in are:

- *User (interactive) delay*, defined as the delay from the request of a segment until the segment is streamed to the user. Clearly, the delay is the sum of the path delay and the waiting time for available end-to-end streaming bandwidth from the streaming parent to the user. We are mainly interested in the average and distribution of user delay for all *served* requests (i.e., excluding those reneged ones).
- *Miss delay*, defined as the user delay for the segments not stored by the home servers (i.e., misses). We are interested in both its average and distribution.
- *Wait probability*, the probability of a miss which, due to network congestion, needs to wait for the available end-to-end network bandwidth for streaming. This is measured as the ratio of the number of requests that cannot be served immediately to the total number of requests.
- *Reneging rate*, due to reneging of a miss after waiting for a certain amount of time (10 s in our simulation). Reneging rate is calculated as the fraction of requests leaving the system without being served.

We compare CR–SS–SP with the following schemes. For all of them, if a request cannot be served directly, the home server will choose the nearest server as the streaming parent:

- *Random*, where each server randomly replicates segments without considering the segment popularity. This is the simplest scheme.
- *MPF (Most Popular First)*, where each server only replicates the most popular segments. It is a greedy approach to maximize local hit.
- *LRU (Least Recently Used)*, where each server only stores the segments that are most recently requested at itself. It is a commonly used scheme.
- *Local Greedy Replication*, where each server independently replaces segments to reduce access cost in accordance with [35]. The access cost takes into

Table 2
Baseline parameters used in the simulation.

Parameter	Default value
Underlay link bandwidth	5 Gbits/s
Number of servers	30
Total storage budget	900
Streaming bitrate	5 Mbits/s
Number of segments	1200
Segment downloading bitrate (for a replacement)	25 Mbits/s
Segment length	5 min
Zipf parameter s	0.4
Request rate for each server	6 per minute
Number of neighbors for each server	15
Duration between two replacement decisions	12 min
Maximum user waiting time	10 s

consideration of both segment access delay and segment popularity. It is a state-of-the-art scheme with cooperative replication.

5.2. Illustrative results

Fig. 2 plots average user delay versus request rate given different schemes. User delay increases with request rate due to increase in network load. CR–SS–SP clearly achieves significantly lower user delay among all the schemes. In other words, given the same requirement on the user delay, CR–SS–SP can support much higher user capacity (i.e., higher requests rate or concurrent users in the system). LRU and Local Greedy do not perform well mainly because of their high frequency of content replacement, which incurs large amount of download traffic in the network. Random replacement does not have any content replacement, and hence achieves lower replacement traffic. MPF, due to its replication of only the most popular segments, does not take advantage of *cooperative* replication. CR–SS–SP achieves the best performance because it takes advantages of cooperative replication, achieving low replacement overhead and better storage planning. Fig. 3 plots the average miss delay versus request rate given different schemes. We have similar observation here with the user delay. CR–SS–SP achieves much lower miss delay as compared with all other schemes due to its better cooperative replication. As the performance of miss delay is qualitatively the same as user delay, we will only show user delay in the following.

Fig. 4 shows the user delay distribution given different schemes. Most of the requests have low user delay (i.e., less than 2 s). CR–SS–SP significantly outperforms other schemes because its replication strategy considers the delay in the network much better. It avoids frequent access to the repository and far-away remote servers, which will bring longer user delay and higher reneging rate. Moreover, it has effective server selection using the probabilistic framework. All these lead to that CR–SS–SP distributes traffic more uniformly over the network, resulting in much fewer bottleneck links as compared with other schemes. It also enjoys the lowest reneging rate among all the schemes.

Fig. 5 plots average user delay versus streaming bitrate given different schemes. User delay increases with the streaming bitrate, because network traffic increases with

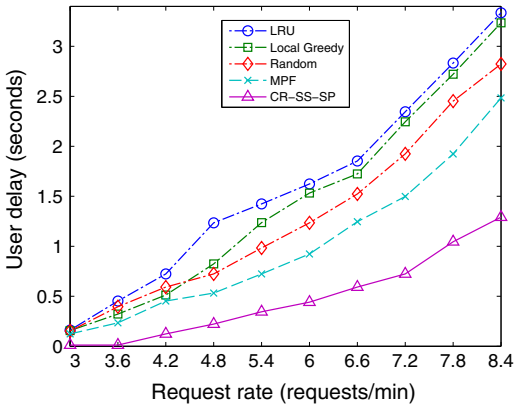

Fig. 2. Average user delay versus request rate given different schemes.

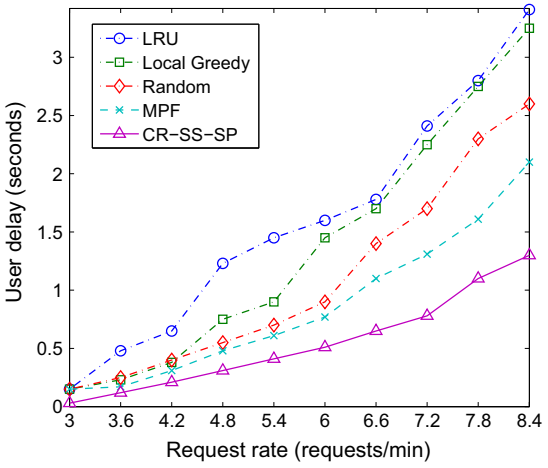

Fig. 3. Average miss delay versus request rate given different schemes.

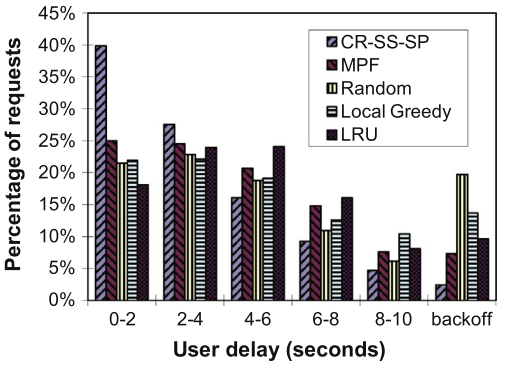

Fig. 4. User delay distribution given different schemes.

streaming bitrate. It is clear that CR–SS–SP achieves significantly the lowest user delay as compared with all other schemes. When the streaming bitrate increases, LRU, Local Greedy and MPF do not perform well. In contrast, CR–SS–SP considers the delays much better which significantly reduces the miss delay. MPF and Random perform better than LRU and Local Greedy due to their lower downloading

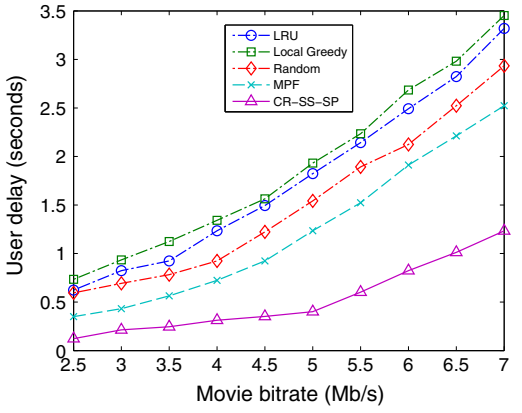

Fig. 5. Average user delay versus streaming bitrate given different schemes.

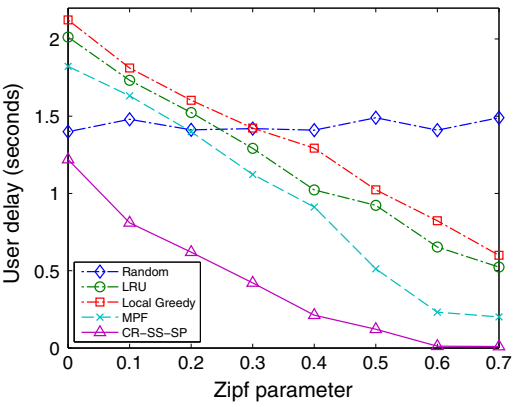

Fig. 7. Average user delay versus Zipf parameter s given different schemes.

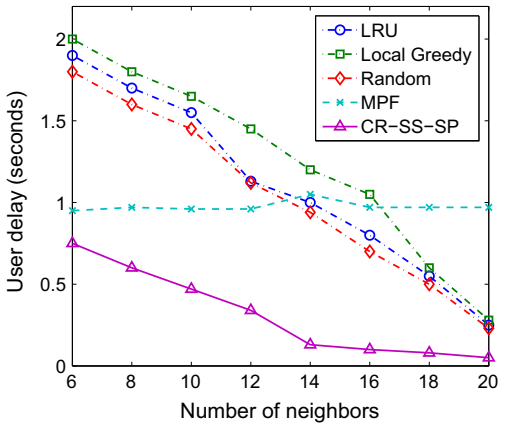

Fig. 6. Average user delay versus number of neighbors given different schemes.

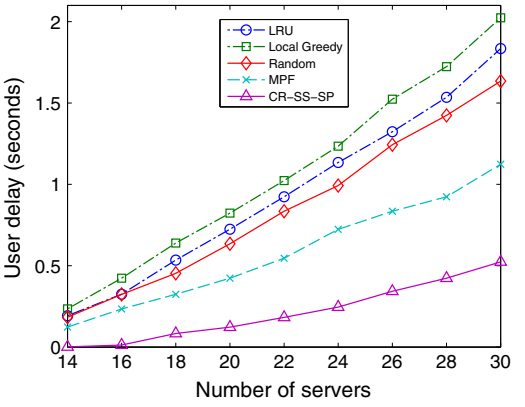

Fig. 8. Average user delay versus number of servers given different schemes.

traffic. Given the same requirement on user delay, CR–SS–SP is able to support much higher streaming bitrate and hence video quality.

Fig. 6 plots average user delay versus the number of neighbors of each server. For MPF, the average user delay does not depend sensitively on the number of neighbors because the servers always replicate the same set of segments, i.e., the most popular segments. For the other schemes, the average user delay decreases with the number of neighbors. This is mainly because SS can be better optimized with larger number of neighbors. CR–SS–SP achieves much lower delay, because its probabilistic server selection and storage planning spreads traffic across the network more effectively.

Fig. 7 plots the average user delay versus Zipf parameter s given different schemes. User delay decreases with the Zipf parameter. This is because skewed popularity means that more requests are concentrated on fewer segments. Consequently, the miss rate decreases, leading to lower delay. CR–SS–SP achieves substantially the lowest delay than the other schemes, even when s is small (i.e., segment popularity is rather uniform). This shows that CR–SS–SP makes very good cooperative replication and storage decisions. When s is large, CR–SS–SP achieves similar performance as MPF. This is because it makes effective decision as MPF by replication high-popularity segments at each of the servers. Random does not perform well, because it does not replace its stored segments according to the segment popularity. Compared with Random, LRU and Local Greedy perform better because they adapt their replication decision according to the segment popularity. However, due to their continuous segment replacement, LRU and Local Greedy generate a lot of traffic which leads to higher user delay as compared with CR–SS–SP.

Fig. 8 plots average user delay versus number of servers in the system. As the number of servers increases, user traffic increases (as each additional server carries user traffic). The increase in user traffic leads to higher user delay. CR–SS–SP performs the best, because its replication strategy, server selection and storage planning can effectively spread the traffic over the network, leading to lower network congestion.

Fig. 9 plots wait probability versus request rate. The probability increases with request rate due to network congestion. CR–SS–SP achieves the lowest wait probability due to its better optimization. LRU and Local Greedy do not perform well because of their ineffective replication and

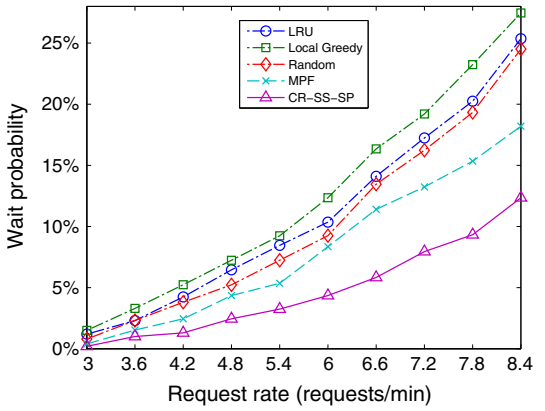

Fig. 9. Average wait probability versus request rate given different schemes.

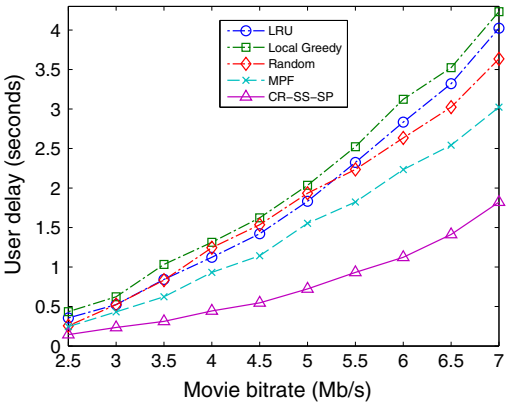

Fig. 12. Average user delay versus streaming bitrate given different schemes on a real ISP topology.

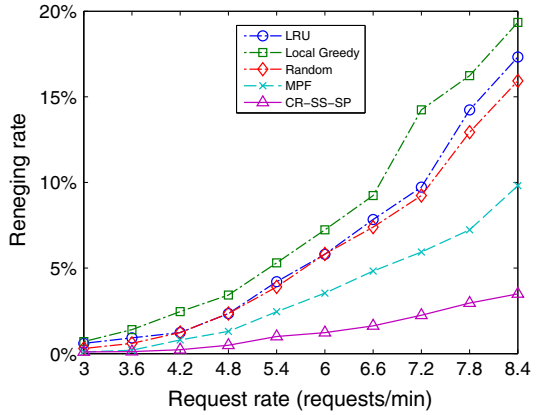

Fig. 10. Average reneging rate versus request rate given different schemes.

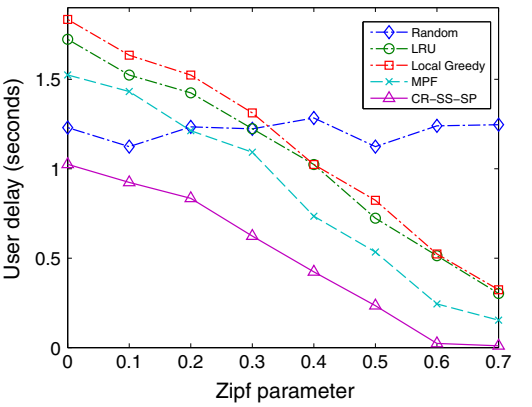

Fig. 13. Average user delay versus Zipf parameter s given different schemes on a real ISP topology.

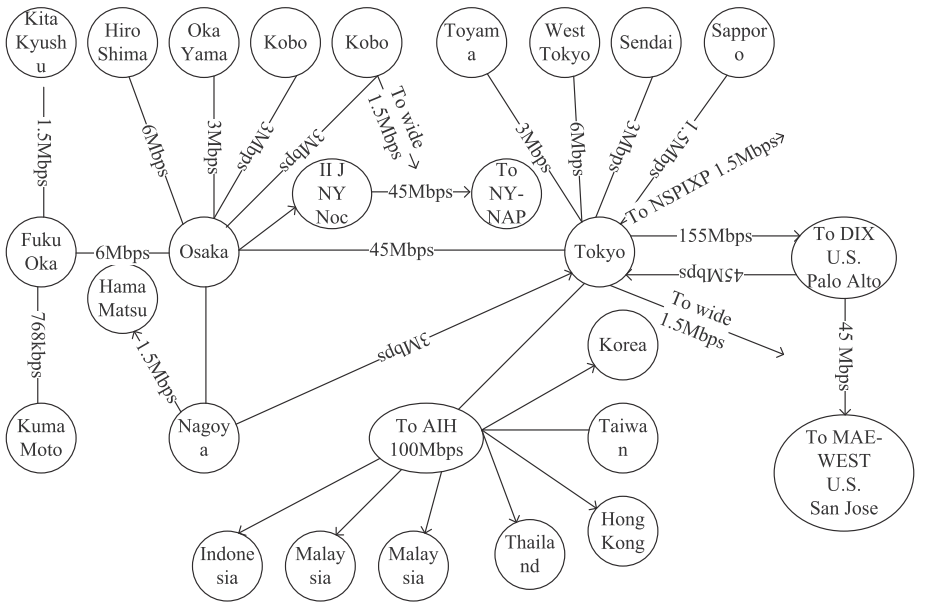

Fig. 11. Topology obtained from an ISP.

server selection algorithms. As the request rate increases, the repository for MPF becomes congested, because the servers are "greedy", and they only replicate the popular segments without cooperating with each other.

Fig. 10 plots the reneging rate versus request rate. The reneging rate increases with request rate due to network congestion. CR–SS–SP has substantially the lowest reneging rate as compared with all the other schemes, for the same reason as discussed before.

Besides Internet-like topologies generated by BRITE, we have also simulated using a realistic ISP network. Fig. 11 shows the topology and the bandwidth of each link as obtained from [43]. The core routers, edge routers and gateways are indicated as circles, small rectangles and larger rectangles, respectively. Servers are randomly attached to the edge routers, where requests arrive. All the links are with certain bandwidth and delay in accordance with the discussion of an ISP. Figs. 12 and 13 plot the effects of the streaming bitrate and Zipf parameter s on user delay, respectively. Note the remarkable resemblance with the previous results based on BRITE simulation (Figs. 5 and 7). After examining many other plots, the results are qualitatively similar to the simulated Internet topologies and the results will not be repeated here.

6. Conclusion

In this work, we study the provisioning of large-scale distributed video-on-demand (VoD) services where cooperative servers are deployed close to user pools. We address the joint optimization of content replication (CR), server selection (SS) and storage planning (SP) to achieve low user (interactive) delay. We formulate an optimization problem, and propose a simple and distributed algorithm called CR–SS–SP to jointly optimize the system. With CR–SS–SP, each server uses a simple routing table to store segment location. By locally exchanging the routing table, each server is able to find good streaming server(s) for any of the segments. It is used in storage planning to allocate server capacity given a certain storage budget. CR–SS–SP makes effective distributed replacement decision to minimize user delay.

We have conducted extensive simulation studies of CR–SS–SP on both Internet-like and realistic topologies. Our results show that our algorithm achieves much better user delay and miss delay comparing to the both traditional and state-of-the-art schemes. We can also see that CS–SS–SP performs well under different segment popularity distributions. It is adaptive and not restricted to a certain movie popularity. CR–SS–SP is able to support by far the highest streaming bitrate and request rate, and achieves the lowest miss delay, probability of waiting, and reneging rate. In the future we will study the joint optimization of traffic engineering (TE) and CR, SS, SP, and see how to improve our system given the underlying ISP traffic information.

References

[1] X. Zhang, H. Hassanein, Video on-demand streaming on the internet – a survey, in: 2010 25th Biennial Symposium on Communications (QBSC), 2010, pp. 88–91.

[2] Z. Chen, C. Lin, X. Wei, Enabling on-demand internet video streaming services to multi-terminal users in large scale, IEEE Trans. Consum. Electron. 55 (2009) 1988–1996.

[3] C. Zheng, Z. Ming, An efficient video similarity search strategy for video-on-demand systems, in: 2nd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT 2009), 2009, pp. 174–178.

[4] J. Choi, M. Yoo, B. Mukherjee, Efficient video-on-demand streaming for broadband access networks, IEEE/OSA J. Opt. Commun. Network. 2 (2010) 38–50.

[5] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, Z.-L. Zhang, Unreeling netflix: understanding and improving multi-CDN movie delivery, in: INFOCOM, 2012 Proceedings IEEE, 2012, pp. 1620–1628.

[6] K.N. Parvez, C. Williamson, A. Mahanti, N. Carlsson, Insights on media streaming progress using BitTorrent-like protocols for on-demand streaming, IEEE/ACM Trans. Network. 20 (2012) 637–650.

[7] Z. Ma, K. Xu, J. Liu, H. Wang, Measurement, modeling and enhancement of BitTorrent-based VoD system, Comput. Networks: Int. J. Comput. Telecommun. Network. 56 (2012) 1103–1117.

[8] T. Zhang, X. Cheng, J. Lv, Z. Li, W. Shi, Providing hierarchical lookup service for P2P-VoD systems, ACM Trans. Multimedia Comput., Commun., Appl. (TOMCCAP) 8s (2012).

[9] S.-H. Chan, F. Tobagi, Distributed servers architecture for networked video services, IEEE/ACM Trans. Network. 9 (2001) 125–136.

[10] H. Febiansyah, J.B. Kwon, Dynamic proxy-assisted scalable broadcasting of videos for heterogeneous environments, Multimedia Tools Appl. 66 (2013) 517–543.

[11] C. Huang, J. Li, K.W. Ross, Can internet video-on-demand be profitable? in: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, USA, 2007, pp. 133–144.

[12] P.R.R.S. Annapureddy, C. Gkantsidis, L. Massoulie, Providing video-on-demand using peer-to-peer networks, in: Microsoft Research Technical Report, MSR-TR-2005-147, October 2005.

[13] W.-P.K. Yiu, X. Jin, S.-H.G. Chan, VMesh: distributed segment storage for peer-to-peer interactive video streaming, IEEE J. Sel. Areas Commun. (JSAC) 25 (2007) 1717–1731 (special issue on Advances in Peer-to-Peer Streaming Systems).

[14] Y. He, Z. Xiong, Y. Zhang, X. Tan, Z. Li, Modeling and analysis of multi-channel P2P VoD systems, J. Network Comput. Appl. 35 (2012) 1568–1578.

[15] M. Hefeeda, B. Noorizadeh, On the benefits of cooperative proxy caching for peer-to-peer traffic, IEEE Trans. Parallel Distrib. Syst. 21 (2010) 998–1010.

[16] J. Dai, Z. Hu, B. Li, J. Liu, B. Li, Collaborative hierarchical caching with dynamic request routing for massive content distribution, in: INFOCOM, 2012 Proceedings IEEE, 2012, pp. 2444–2452.

[17] B. Tan, Optimal content placement for peer-to-peer video-on-demand systems, IEEE/ACM Trans. Network. 21 (2013) 566–579.

[18] L. Qiu, Y.R. Yang, Y. Zhang, S. Shenker, On selfish routing in internet-like environments, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2003, pp. 151–162.

[19] Y. Zhu, C. Dovrolis, M. Ammar, Dynamic overlay routing based on available bandwidth estimation: a simulation study, Comput. Networks 50 (2006) 742–762.

[20] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann, B. Maggs, Enabling content-aware traffic engineering, SIGCOMM Comput. Commun. Rev. 42 (2012) 21–28.

[21] E. Koutsoupias, C. Papadimitriou, Worst-case equilibria, Comput. Sci. Rev. 3 (2009) 65–69.

[22] P. Xia, S.-H.G. Chan, M. Chiang, G. Shi, H. Zhang, L. Wen, Z. Yan, Distributed joint optimization of traffic engineering and server selection, in: Packet Video Workshop (PV), 2010 18th International, 2010, pp. 86–93.

[23] Y.R. Choe, D.L. Schuff, J.M. Dyaberi, V.S. Pai, Improving VoD server efficiency with bittorrent, in: Proceedings of the 15th International Conference on Multimedia, New York, NY, USA, 2007, pp. 117–126.

[24] Y. Zhou, T.Z.J. Fu, D.M. Chiu, On replication algorithm in p2p vod, IEEE/ACM Trans. Network. (TON) 21 (2013) 233–243.

[25] J. Lv, X. Cheng, Q. Jiang, J. Ye, T. Zhang, S. Lin, L. Wang, LiveBT: providing video-on-demand streaming service over bittorrent

systems, in: International Conference on Parallel and Distributed Computing Applications and Technologies, 2007, pp. 501–508.
[26] Y. Huang, T.Z. Fu, D.-M. Chiu, J.C. Lui, C. Huang, Challenges, design and analysis of a large-scale P2P-VoD system, in: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication, 2008, pp. 375–388.
[27] N. Carlsson, D.L. Eager, Server selection in large-scale video-on-demand systems, ACM Trans. Multimedia Comput., Commun., Appl. 6 (2010) 1–26.
[28] D. DiPalantino, R. Johari, Traffic engineering vs. content distribution: a game theoretic perspective, in: INFOCOM 2009, IEEE, 2009, pp. 540–548.
[29] W. Jiang, R. Zhang-Shen, J. Rexford, M. Chiang, Cooperative content distribution and traffic engineering in an ISP network, in: Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '09, 2009, pp. 239–250.
[30] S. Sivasubramanian, M. Szymaniak, G. Pierre, M. van Steen, Replication for web hosting systems, ACM Comput. Surv. (CSUR) 36 (2004).
[31] J.-B. Chen, C.-C. Chen, Using particle swarm optimization algorithm in multimedia CDN content placement, in: Parallel Architectures, Algorithms and Programming (PAAP), 2012 Fifth International Symposium on, 2012, pp. 45–51.
[32] M. Al-Shayeji, S. Rajesh, M. Alsarraf, R. Alsuwaid, A comparative study on replica placement algorithms for content delivery networks, in: Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on, 2010, pp. 140–142.
[33] H.-S. Tang, S.-H.G. Chan, H. Li, Optimizing segment caching for peer-to-peer on-demand streaming, in: Proceedings of the 2009 IEEE International Conference on Multimedia and Expo, 2009, pp. 810–813.
[34] A.T.S. Ip, J. Liu, J.C.-S. Lui, COPACC: an architecture of cooperative proxy-client caching system for on-demand media streaming, IEEE Trans. Parallel Distrib. Syst. 18 (2007) 70–83.
[35] S. Borst, V. Gupta, A. Walid, Self-organizing algorithms for cache cooperation in content distribution networks, ACM SIGMETRICS Perform. Eval. Rev. 37 (2009) 71–72.
[36] H. Luss, Optimal content distribution in video-on-demand tree networks, IEEE Trans. Syst., Man Cybernet., Part A: Syst. Hum. 40 (2010) 68–75.
[37] H. Huang, P. Xia, S.-H.G. Chan, G. Shi, H. Zhang, Joint optimization of distributed replication and server selection for video-on-demand, in: Proceedings of IEEE International Conference on Communications (ICC), 2012.
[38] U.C. Kozat, O. Harmanci, S. Kanumuri, M.U. Demircin, M.R. Civanlar, Peer assisted video streaming with supply-demand-based cache optimization, IEEE Trans. Multimedia 11 (2009) 494–508.
[39] D. Gross, C.M. Harris, Fundamentals of Queueing Theory (Wiley Series in Probability and Statistics), Wiley-Interscience, 1998.
[40] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.
[41] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, third ed., 2009.
[42] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, first ed., 1979.
[43] Cybergeography, 2007. <http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography>.

Dongni Ren received his B.Eng. and M.Phil. degrees in Computer Science from the Hong Kong University of Science and Technology (HKUST), Kowloon, in 2007 and 2009. He is currently pursuing the Ph.D. degree at the Department of Computer Science and Engineering, HKUST. His research interests include multimedia networking and overlay live streaming.

S.-H. Gary Chan (S'89-M'98-SM'03) is currently Full Professor and Undergraduate Programs Coordinator at the Department of Computer Science and Engineering and Director of Sino Software Research Institute, The Hong Kong University of Science and Technology (HKUST), Hong Kong. He received MSE and PhD degrees in Electrical Engineering from Stanford University (Stanford, CA) in 1994 and 1999, respectively, with a minor in business administration. He obtained his B.S.E. degree (highest honor) in Electrical Engineering from Princeton University (Princeton, NJ) in 1993, with certificates in Applied and Computational Mathematics, Engineering Physics, and Engineering and Management Systems. His research interest includes multimedia networking, overlay streaming, wireless networks and mobile applications. Professor Chan was an Associate Editor of IEEE Transactions on Multimedia (2006–2011), and is a Vice-Chair of Peer-to-Peer networking and Communications Technical Sub-Committee of IEEE Comsoc Emerging Technologies Committee. He has been Guest Editors of IEEE Transactions on Multimedia (2011), IEEE Signal Processing Magazine (2011), IEEE Communication Magazine (2007), and Springer Multimedia Tools and Applications (2007). He was the TPC chair of IEEE Consumer Communications and Networking Conference (CCNC) 2010, Multimedia symposium in IEEE Globecom (2007 and 2006), IEEE ICC (2007 and 2005), and Workshop on Advances in Peer-to-Peer Multimedia Streaming in ACM Multimedia Conference (2005). Professor Chan's research projects on wireless and streaming have received several ICT (Information and Communication Technology) award sin Hong Kong, Pan Pearl River Delta and Asia–Pacific regions due to their commercial impacts to industries (2012 and 2013). He is the recipient of Google Mobile 2014 Award (2010 and 2011) and Silver Award of Boeing Research and Technology (2009). He has been a visiting professor or researcher in Microsoft Research (2000–2011), Princeton University (2009), Stanford University (2008–2009), and University of California at Davis (1998–1999). He was a Co-director of HKUST Risk Management and Business Intelligence program (2011–2013), and Director of Computer Engineering Program at the HKUST (2006–2008). He was a William and Leila Fellow at Stanford University (1993–1994), and the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence at Princeton (1993).

Guangyu Shi is currently Distinguished Engineer at Huawei. He received his bachelor degree from Tsinghua University, China, and master degree from Dalian Technology University, China. He was the leader of iSAP TDT, and Vice Technical Director at Huawei. His research focuses on Distributed Computing, Storage and Networking.

Zhang Hongbo received his Ph.D from Huazhong University of Science and Technology, after that he joined Huawei as a researcher who mainly focused on cross-layer network joint optimization (includes joint optimization among traffic engineering, server selection and cooperative cache) to optimize network performance. He is now senior researcher at Huawei, and his current research interests are around big data analysis and processing field, including how to design advanced computing platforms by using parallel/distributed computing techniques to support big data analysis by using advanced machine learning algorithms, recommender systems, etc.