IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 10, 2018, accepted March 14, 2018, date of publication March 21, 2018, date of current version April 23, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2817646

Joint Replica Server Placement, Content Caching, and Request Load Assignment in Content Delivery Networks

KAI XU[1,2], XIANG LI[1], SANJAY KUMAR BOSE[3], (Senior Member, IEEE), AND GANGXIANG SHEN[1], (Senior Member, IEEE)

[1]School of Electronic and Information Engineering, Soochow University, Suzhou 215006, China
[2]Department of Electronic Engineering, City University of Hong Kong, Hong Kong
[3]Department of EEE, IIT Guwahati, Guwahati 781039, India

Corresponding author: Gangxiang Shen (shengx@suda.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61671313 and Grant 61322109 and in part by the Science and Technology Achievement Transformation Project of Jiangsu Province, China, under Grant BA2016123.

ABSTRACT With the explosive growth of information and communication technology and its services, some popular Websites currently generate an enormous amount of Internet traffic. A content delivery network (CDN) would then become imperative for supporting such services efficiently. In this paper, we propose joint optimizing approaches for replica server placement, content caching in selected servers, and content request load assignment among the servers, aiming to minimize the ratio of unserved content request load when the network resources and server capacity are both limited. For this, we develop a mixed integer linear programming (MILP) optimization model. To mitigate the computational complexity of the MILP model, we further decompose the optimization problem into three sub-problems, including: 1) choosing the replica server nodes optimally; 2) deciding the content items to be cached in the replica servers; and 3) allocating the content request loads from users onto different servers. For these sub-problems, we develop corresponding heuristic algorithms and show that the proposed approach is not only efficient but also performs very close to the MILP model. We also find that a number of system limitations, such as different numbers of replica servers placed, link capacity, server processing capacity, and server storage capacity, jointly affect the performance of the CDN. Saturation trends are observed on the performance, which indicate that as long as sufficient resources have already been provided, augmenting the resources further may only lead to marginal additional performance improvement.

INDEX TERMS Content delivery network, replica server placement, content caching, network resource allocation.

I. INTRODUCTION

With the increasing expansion of Internet usage and high volume applications, some popular websites now generate very high Internet traffic. Cisco predicted that, globally, 71% of all Internet traffic will go through CDNs by 2021, going up from the 52% figure in 2016 [1]. In this scenario, offering good Quality of Service (QoS) to users at a low cost becomes a challenging issue for these websites motivating an increasing trend towards CDNs [2], [3].

The main feature of CDNs is the deployment of replica servers and caching of popular content nearer to the users for faster content access. This not only reduces request latency but also makes it possible to balance the load between the content servers [4]. Appropriate placement of replica servers can significantly reduce the transmission latency of requested content and will also reduce its bandwidth consumption. Moreover, given that the servers have limited storage and processing capacity, a suitable strategy is needed to choose the content items to be cached in a replica server as that would affect the performance of the CDN. Thus, a high overall efficiency is achieved for a CDN when it is able to deliver content with high performance (quantified as reduced latency or a strict bound on QoS) and low cost [4]. Achieving these two goals influences the key design problems of CDNs.

A general statement of the replica server placement problem would be as follows. Given a set of candidate locations (or nodes) for replica server placement and a set of distributed end users, the problem is to find the optimal locations for replica server placement from the given candidate set so that end-users' quality of experience (QoE) and the overall performance of a CDN is the best. This would be subject to the constraints on various parameters, such as the processing capacity of the replica servers, the bandwidth of the network links, and the QoS requirement of the end-users. The replica server placement problem is a type of K-center (NP-hard) problem, for which various heuristic algorithms have been proposed in the literature [5]–[10]. We will give a more detailed survey on this in the next section.

In a similar way, a general statement of the content caching problem would be as follows. For the numerous content items provided in the network, each content has different popularities for different users. The problem then is to decide which content items should be cached in each replica server so that the overall average delivery latency of the content is minimized. The key constraint for this problem is the limited storage capacity of each replica server. A typical content caching strategy is to cache the most popular content in the replica servers. As a side-benefit, an efficient content caching strategy can also help to reduce the bandwidth consumed for data delivery from the origin server. In the literature, many strategies have been proposed for content caching in the replica servers [11]–[20]. We will give a more detailed survey on this in the next section.

We see that most of the existing studies on CDN optimization focus on either replica server placement or content caching schemes, but do not jointly optimize both objectives. In this paper, as a key novel aspect, we jointly consider the *three* issues of *replica server placement* and *content caching* along with *content request load assignment* onto different servers. Subject to a limited number of replica servers to be placed, limited network link bandwidth, limited storage and processing capacity of each replica server, we jointly maximize the amount of content request load served while minimizing the average content delivery latency. As key contributions, we propose a Mixed Integer Linear Programming (MILP) formulation and subsequently propose efficient heuristic algorithms to determine the locations of replica servers placed, the content to be cached in each replica server, and the algorithm for content request load assignment. Specifically, a *Server List Growing* (SLG)-based replica-server placement algorithm, a *User Visiting Popularity* (UVP)-based content caching algorithm, and a *Server-based Closest First* (CF) content load request assignment algorithm are developed, respectively. Simulation results show that the heuristic algorithms are effective in maximizing the amount of content request load served and reducing the average delivery latency in the whole network and that they jointly perform close to the optimal results obtained by the MILP model.

The rest of the paper is organized as follows. Section II provides a literature survey on the current studies of CDNs. In Section III, we introduce the relevant concepts of CDN and illustrate the benefit of using them to improve network resource utilization. For joint replica server placement, content caching in the replica servers, and content request load assignment, we develop an MILP model and heuristic algorithms in Sections IV and V, respectively. In Section VI, we evaluate the performance of these approaches for some example scenarios and the corresponding results are presented and discussed. Section VII concludes the paper.

II. RELATED WORK

Content Delivery Networks (CDNs) have drawn considerable research interest recently. An overview of CDNs is presented in [2], which describes the current CDN ecosystem and the forces that have driven its evolution. It also describes different CDN architectures and their relative strengths and weaknesses. The growing complexity of the CDN ecosystem is illustrated and its implications for interconnection markets are discussed. A survey on peer-assisted CDNs has also been made, focusing on its remarkable potential for reducing the burden of user requests on content delivery servers [3]. The authors reviewed and systematized the ongoing debate around the future of peer-assisted networks and proposed a novel taxonomy to characterize the research and industrial efforts in this area. In the context of Cloud based Content Delivery Networks (CCDNs), a comprehensive survey of content placement algorithms for CCDNs was provided in [21].

Replica server placement and content caching are two key problems in a CDN. In general, the replica-server placement problem is NP-hard, for which various optimal and heuristic algorithms have been proposed. A comprehensive survey on replica server placement algorithms for CDNs was reported in [5]. The authors reviewed the replica server placement algorithms in traditional and emerging paradigm-based CDNs, identified the requirements for an efficient replica server placement algorithm and performed a comparison in the light of these requirements. They also discussed several potential avenues for further research in replica server placement in CDNs. A similar comprehensive survey of replica server placement algorithms is also given in [6], which discusses a number of algorithms and compares their performance based on various optimization factors.

Several studies on the replica server placement problem have been carried out recently in the context of particular networks. For an elastic optical network (EON), a study on joint content placement and lightpath routing and spectrum assignment for CDNs was done in [7]. This used an Integer Linear Programming (ILP) formulation to solve optimally the targeted problem and subsequently used a heuristic approach called CPRMSA-PD to decompose the optimization problem into three sub-problems for better tractability. An efficient greedy heuristic for the replica server placement problem was proposed in [8]. This consisted of various placement and refinement steps to ensure an efficient placement for the CDN servers. Similar studies have also been reported for virtual

CDNs [9] and for CDNs with edge-provisioning and flexible server placement [10].

The content caching problem is to decide the content to be cached at each replica server subject to a limited server storage capacity. For this, many strategies and schemes have been proposed in the literature [11]–[20]. In [11]–[13], the performance of content caching strategies were evaluated in simple cascaded or tree topologies. Borst *et al.* [11] developed a lightweight cooperative cache management algorithm, aiming to maximize the traffic volume served by the cache and to minimize the bandwidth cost. A content caching scheme was also proposed in [12], in which the number of content chunks to be cached was adjusted based on the popularity of the content. Kim and Yeom [13] formulated the optimal content assignment for two caching policies, i.e., *Single-Path Caching* and *Network-Wide Caching*, where the objectives were to maximize the cache-hit ratio and to minimize the average response time.

Recently, studies on content caching have been further extended to wireless networks because of the increasing popularity and capacity of data networks using mobile telephony, e.g., LTE and 5G networks [14]–[20]. For example, Sung *et al.* [22] proposed an ILP model and a cross-layer heuristic algorithm to deal with the problem of cache placement in a two-tier wireless content delivery network (WCDN). Also, Sung *et al.* [23] modelled the content caching problem as a Markov Decision Process and applied reinforcement learning to the content caching problem in a WCDN. In [14], a cluster content caching structure was proposed for cloud radio access networks (C-RANs) to resolve the issues of potential high power consumption and poor quality of service (QoS) experience for real-time services. Kanai *et al.* [15] proposed a proactive content caching scheme that used actual transportation systems. This scheme was tested to deliver video streaming through two field experiments using actual trains and was found to be quite effective. The content caching technique has also been applied to the heterogeneous cellular network (HetNet), which is a promising architectural technique for 5G mobile networks [16]. Here, the authors developed an optimal cooperative content caching and delivery policy, where the femto base-stations (FBSs) and user equipments (UEs) were cooperatively engaged in local content caching. Numerical results showed that the proposed policy could significantly improve content delivery performance in comparison with existing caching strategies. Other similar studies on content caching for the wireless networks can also be found in the literature of [17]–[20].

Based on the above literature survey, we summarize that most of the existing studies on CDN optimization focus either on replica server placement, on content caching schemes, or on content request load assignment. However, these three sub-problems were tackled separately, but not jointly. Only a few of the studies have focused on joint optimization for the above problems.

Joint optimization for two of the three sub-problems mentioned above has been considered in the study of [24], which tackled the joint problem of replica placement and content request distribution. Laoutaris *et al.* [25] proposed ILP models and heuristic algorithms to jointly solve the problem of replica server placement and content caching. In [26], a near-optimal solution of replica server placement and content request load assignment was proposed. Bektas *et al.* [27] described two exact algorithms for the joint problem of replica server placement and content request load assignment in a CDN. In [28], the problem of replica replacement and request distribution was solved by a hybrid method. An optimization scheme and a two-stage algorithm were proposed in [29] to jointly solve the problem of content caching and content request load assignment in CCDNs.

As for the joint optimization of the three sub-problems, Bektas *et al.* [30] provided ILP models and heuristic algorithms to jointly solve the problem of cache deployment, content request routing, and non-cooperative content replication. The work in [31] was an extension of [30], where cooperation among caching servers was also considered. Both [30] and [31] aimed at minimizing the cache deployment cost and the content transmission cost. Unlike [30] and [31], in our work we focus on multi-objective optimization for the CDN deployment, aiming to simultaneously maximize the hit ratio and minimize the delivery latency. This is done by having the primary objective be to maximize the total amount of satisfied content request load in the whole network. We considered the limited storage capability and processing capacity of replica servers as constraints, while [31] only invoked the constraint of storage capacity in their model. In [31], compared to the anycast technique applied by us to serve user requests, the user requests can only be served by the selected one of many candidates from its local replica server, original server, and neighboring cache servers of the local replica server.

In summary, as the key novel aspect, our study focuses on the approach of *jointly* optimizing the problems of replica server placement, content caching, and content request load assignment. For this, an MILP model for the joint optimization is formulated and an efficient heuristic approach that jointly incorporates the above three sub-problems is developed. The proposed heuristic approach is simulated and is observed to perform very close to the optimal performance achieved by the MILP model.

III. RELEVANT CONCEPTS OF CDN

In this section, some of the important concepts relevant to this work are introduced with examples.

A. CONTENT DELIVERY NETWORK AND ANYCAST

We use the example in Fig. 1 to illustrate a CDN network, where five network (switching) nodes make up a communication network for content delivery. An origin server is located at node 4 with two replica servers placed at nodes 0 and 3, respectively. At each network node, a group of content users is attached, which always request their content items from the closest servers via this network node. The origin server would

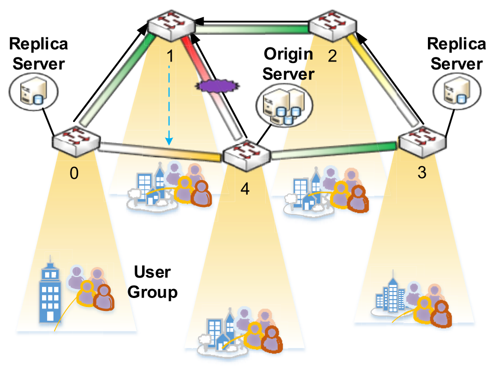


FIGURE 1. An example of content delivery network (CDN).

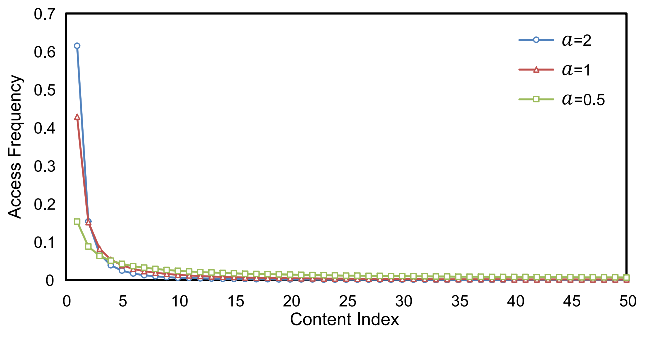


FIGURE 2. An example of the Zipf distribution.

host all of the content items. This can be a mega-datacenter and, in this study, we assume that it has very high storage and processing capacity. To allow users to access content items fast with less network overhead, some content items are duplicated in the replica servers. However, these replica servers have limited storage and processing capacities, which means that they can only store a limited number of content items and can only serve a limited number of users simultaneously. In this context, if a server and a user are directly attached to the same network node, we call the server the *local server* for the user; otherwise, the server is called the *remote server* for the user. In Fig. 1, the replica server attached to node 0 is a local server for all the users in the group of node 0, but this server is considered as a remote one for the users in the groups of other nodes from 1 to 4.

In order to efficiently access content, the anycast technique is employed in the CDN. When multiple replica servers host a specific content item, the anycast technique chooses the server that is the closest to the user to deliver it. This technique not only reduces the content delivery latency, but also helps to balance the load among the servers to save network bandwidth and improve the content hit ratio in the CDN.

Fig. 1 illustrates an example of how anycast can enhance content delivery efficiency. Assume that a specific content item is stored in the origin server at node 4. Now assume that there is a new request for this content item from a user attached to node 1. Because link 1-4 (in red) is suffering from a heavy traffic load, it may not have enough capacity to deliver the content item directly from node 4 to node 1. However, by enabling the anycast technique, we can let the replica servers at nodes 0 and 3 both cache the content, and then choose one of them to serve the request, thereby both improving the content hit ratio and balancing the load on the servers.

B. ZIPF's DISTRIBUTION

In a CDN, the popularity of a particular content (i.e., User Visiting Popularity, UVP) is correlated with the frequency with which it is requested. Studies on user content access patterns have found that the UVP follows the *Zipf Distribution* as shown in Fig. 2 [32], [33]. Among v content items being distributed, the UVP of item i is indicated by its access frequency (probability) C/i^{α}, where the parameter C is a normalization constant satisfying $\sum_{i=1}^{v} \frac{C}{i^{\alpha}} = 1$ and α is the distribution parameter. A larger α implies a greater difference in the user access pattern with more users requesting popular items that are only a small fraction of the content supplied in the network. On the contrary, a smaller α means that the difference in the user access patterns is smaller.

From the Zipf distribution, it is easy to see that in a CDN, caching a popular item in a replica server will significantly improve the hit ratio at the server and that this, in turn, will reduce the overall content delivery latency for the users. As the example in Fig. 2 indicates, when the distribution parameter α is 2, we only need to cache the most popular 2% of the content items in a replica server to ensure that 70% of the user requests can be served by the replica server.

IV. JOINT OPTIMIZATION OF REPLICA SERVER PLACEMENT, CONTENT CACHING, AND REQUEST LOAD ASSIGNMENT

A. PROBLEM STATEMENT

The joint optimization of replica server placement, content caching, and content request load assignment for the CDN involves jointly choosing appropriate locations in the network for placement of the replica servers, deciding the content that will be cached in the replica servers, and distributing the content request load onto different servers. This is done so that the CDN can achieve the best performance from the perspectives of user experience such as the hit ratio and the content delivery latency.

The given parameters of the problem include the following. (1) the distribution of user content requests which statistically depends on users' preference for the different content, (2) the number of node locations for replica server placement, (3) the capacity of each replica server, which includes its (CPU) processing capacity and storage size, (4) link capacities and transmission delays, and (5) the latency for a server to process a content request.

The objective of the optimization problem is to jointly maximize the user content requests served (i.e., the hit ratio) and to minimize the content delivery latency. These two objectives are just the two QoE requirements of the end users.

The optimization problem is subject to the different constraints as follows. First, the total number of nodes deployed with replica servers is limited. Second, the replica servers placed at a node can only provide a limited processing capacity and content storage size (or caching size). Third, the link capacities in the network are limited and the cumulative content bandwidth required should not exceed these capacities.

In this problem, since the total number of replica servers to be placed is limited, it is important to effectively distribute them around the network according to the distribution of users. Moreover, since the storage sizes of the replica servers are limited (compared to that of the origin server), it is important to optimally choose the content items to be cached based on the content UVPs. Since the processing capacity of each server and the transmission capacity of each network link are limited, it is also important to efficient distribute the content request load onto the different servers. A good choice of these will significantly improve the hit ratio of user requests and reduce the delivery latency experienced by the users if these popular content items can be made available in replica servers closer to the end users and the request loads are properly allocated between the different servers. Generally, this joint optimization would be NP-hard, for which various heuristic approaches would be desirable.

B. MILP MODEL

We consider a network topology $G(N, E)$, where N is a set of nodes and E is a set of bi-directional network links. We assume that the user content requests at each node in the network are known a priori. As described before, the requests from each user for content follow the Zipf distribution, which forms a content load matrix of this user. The popularity of each content item for a user is also measured by the UVP parameter where the sum of the UVP for all the content items is unity. In addition, the origin server is assumed to be pre-placed at a certain location in the network, as decided by the content provider. This origin server is assumed to have a very large storage capacity so that it can host all the contents that may be requested in the network.

We next present the MILP optimization model for the above optimization problem with sets, parameters, and variables given as follows.

Sets:

$\boldsymbol{N}$	The set of network nodes, which corresponds to all the switching nodes as in Fig. 1.
$\boldsymbol{U}$	The set of users (or user nodes) that initiate content requests, which include all the users in the groups that are attached to all the switching nodes as in Fig. 1.
$\boldsymbol{C}$	The set of content items.
$\boldsymbol{N_i}$	The set of all the neighboring nodes of node i in the network.
$\boldsymbol{O}$	The set of network nodes that are deployed as the origin servers. In this study, without losing generality, we assume that there is only a single origin server, so this set contains only one node.

Parameters:

A	The total number of nodes that are placed with replica servers in the network.
c_y	The size of content item y, which is a random number distributed within a certain range.
t_{ij}	The delay for a content item to go through physical link (i, j), which includes the delays for data switching and transmission at the switching node and the propagation delay along the link.
Q	The processing delay for each delivered content item at a content server.
p_y^u	The predicted UVP of content item y for user u. We assume that we know this popularity for each user. However, getting this predicted value would be quite challenging and would be another interesting topic, which is however beyond the scope of the current paper. We also use this term to represent the (statistical) normalized load of content item y from user u, i.e., $\sum_{y\in \boldsymbol{C}} p_y^u = 1$, which means that each user generates one unit of content request load at any moment and p_y^u then corresponds to the probability that the requested content item is y.
G	The maximum normalized processing capacity of each replica server. Here, we assume that each content item consumes one unit of normalized processing capacity and that all the replica servers have the same processing capacity.
G_O	The maximum normalized processing capacity of the origin server. In general, the processing capacity of the origin server would be much higher than that of a replica server.
F	The maximum normalized transmission capacity of each network link. Here, we assume that satisfying each content item request would consume one unit of normalized transmission capacity on each link and that all the network links have the same transmission capacity.
S	The maximum normalized storage capacity at each replica server. Here, we assume that each content item y consumes c_y units of normalized storage capacity and that all the replica servers have the same storage capacity.
$\boldsymbol{\Delta}$	A big value.

Variables:

R_v	A binary variable which is 1 if a server (either a replica server or an origin server) is placed at this node v; otherwise, 0.
β_v^y	A binary variable which is 1 if content item y is cached at server v; otherwise, 0.
$\theta_y^{u,v}$	A variable to denote the normalized load of content item y delivered from server node v to user u.
$\theta_{i,j,y}^{u,v}$	A variable to denote the normalized load of content item y delivered from server node v to user u, which traverses link (i, j).

The objective function and the constraints are formulated as follows.

Objective:

$$\text{Maximize} \sum\nolimits_{u\in \boldsymbol{U},v\in \boldsymbol{N},y\in \boldsymbol{C}} \theta_y^{u,v} - \alpha \cdot \Big(\sum\nolimits_{u\in \boldsymbol{U},v\in \boldsymbol{N},y\in \boldsymbol{C},i\in \boldsymbol{N}\cup \boldsymbol{U},j\in \boldsymbol{N_i}} \left(t_{ij}\cdot \theta_{i,j,y}^{u,v}\right) + \sum\nolimits_{u\in \boldsymbol{U},v\in \boldsymbol{N},y\in \boldsymbol{C}} (\theta_y^{u,v}\cdot Q)\Big) \quad (1)$$

Subject to:

$$\sum\nolimits_{j\in \boldsymbol{N_i}} \theta_{i,j,y}^{u,v} - \sum\nolimits_{j\in \boldsymbol{N_i}} \theta_{j,i,y}^{u,v} = \begin{cases} -\theta_y^{u,v}, & i=u \\ \theta_y^{u,v}, & i=v \\ 0, & otherwise \end{cases} \quad (2)$$

$$\forall i\in \boldsymbol{N}\cup \boldsymbol{U},\quad \forall u\in \boldsymbol{U},\ \forall y\in \boldsymbol{C},\ \forall v\in \boldsymbol{N}$$

$$\sum\nolimits_{y\in \boldsymbol{C}} \beta_v^y \cdot c_y \le S \quad \forall v\in \boldsymbol{N}\&\&v\notin \boldsymbol{O} \quad (3)$$

$$\beta_v^y = 1 \quad \forall y\in \boldsymbol{C},\ \forall v\in \boldsymbol{O} \quad (4)$$

$$\theta_y^{u,v} \le \Delta\cdot\beta_v^y \quad \forall u\in \boldsymbol{U},\ \forall v\in \boldsymbol{N},\ \forall y\in \boldsymbol{C} \quad (5)$$

$$\theta_y^{u,v} \le \Delta\cdot R_v \quad \forall u\in \boldsymbol{U},\ \forall v\in \boldsymbol{N},\ \forall y\in \boldsymbol{C} \quad (6)$$

$$\beta_v^y \le R_v \quad \forall v\in \boldsymbol{N},\ \forall y\in \boldsymbol{C} \quad (7)$$

$$\sum\nolimits_{v\in \boldsymbol{N}} \theta_y^{u,v} \le p_y^u \quad \forall u\in \boldsymbol{U},\ \forall y\in \boldsymbol{C} \quad (8)$$

$$\sum\nolimits_{u\in \boldsymbol{U},v\in \boldsymbol{N},y\in \boldsymbol{C}} \theta_{i,j,y}^{u,v} \le F \quad \forall i\in \boldsymbol{N},\ \forall j\in \boldsymbol{N_i} \quad (9)$$

$$\sum\nolimits_{u\in \boldsymbol{U},y\in \boldsymbol{C}} \theta_y^{u,v} \le G_O \quad \forall v\in \boldsymbol{O} \quad (10)$$

$$\sum\nolimits_{u\in \boldsymbol{U},y\in \boldsymbol{C}} \theta_y^{u,v} \le G \quad \forall v\in \boldsymbol{N}\&\&v\notin \boldsymbol{O} \quad (11)$$

$$\sum\nolimits_{v\in \boldsymbol{N}} R_v \le A + |\boldsymbol{O}| \quad (12)$$

$$R_v = 1 \quad \forall v\in \boldsymbol{O} \quad (13)$$

In this model, the primary objective is to maximize the total amount of content request load that is satisfied in the whole network. We would also be interested to find the total content delivery latency in the network as that would be important for the users' perception of the performance quality of the content delivery network and minimizing this would then be the secondary objective of our optimization. For this multi-objective optimization, the parameter α is set to a small value (i.e., 0.0001) to ensure that the primary objective (i.e., request satisfaction) has a higher priority than the secondary objective (i.e., delivery latency).

Constraint (2) ensures flow conservation for the content request load between users and content servers in the network. Constraint (3) is to ensure that the limit on the normalized storage size of each replica server is satisfied, i.e., to ensure that the total amount of content cached in the replica server does not exceed its maximum normalized storage size. Constraint (4) indicates that the origin server caches all of the contents provided in the network and constraint (5) ensures that server v can serve the user requests for content item y only when y is cached in the server. Constraint (6) ensures that a content request can be served by a server at node v only if node v is deployed as a replica server, while constraint (7) ensures that content can be cached at server v only if node v is deployed with a content server. Constraint (8) ensures that the total amount of request load of content item y for user u, as served by different servers, never exceeds the request load of this content item from this user, which is normalized as the content item's UVP of this user. Constraint (9) ensures that the total content traffic (or load) traversing link (i, j) should not exceed the maximum normalized transmission capacity of that network link. Constraints (10) and (11) ensure that the total amount of content request load served by replica (origin) server v should not exceed its maximum normalized processing capacity. Constraint (12) ensures that the number of replica servers deployed in the network should not exceed the maximum number A given. Finally, constraint (13) defines the location of the origin server, which is given and predefined by the content provider.

The complexity of the above MILP model is as follows: the dominant numbers of variables is $\mathrm{O}\left(|\boldsymbol{N}|^3\cdot|\boldsymbol{U}|\cdot|\boldsymbol{C}|\right)$ due to $\theta_{i,j,y}^{u,v}$ and the dominant number of constraints is $\mathrm{O}\left(|\boldsymbol{U}|^2\cdot|\boldsymbol{C}|\cdot|\boldsymbol{N}|+|\boldsymbol{N}|^2\cdot|\boldsymbol{C}|\cdot|\boldsymbol{U}|\right)$ due to constraint (1), where $|\boldsymbol{N}|$ is the total number of candidate nodes for replica server placement, $|\boldsymbol{U}|$ is the total number of users, and $|\boldsymbol{C}|$ is the total number of content items.

V. HEURISTIC APPROACHES

The MILP model will find an optimal solution to the joint problem of replica server placement and content caching. However, since the problem is NP-complete, a long computational time may be required. For better tractability, it would be desirable to develop an efficient heuristic algorithm for this problem as proposed here. Our heuristic algorithm consists of three key steps. These are (i) determining the locations of the replica servers subject to the total number of replica servers to be deployed in the network, (ii) choosing the content items to be cached in each of the replica servers subject to a limited storage capacity at each server, and (iii) assigning content request load to different servers for users subject to the limited processing capacity of each server and limited network link capacity. Note that these three steps are not independent, but would depend on each other for a proper joint optimization. For example, for placing replica servers in step (i), we would consider whether the placement can achieve the best performance for carrying users' content request load in step (iii). Next, we introduce each of the steps for the algorithm.

A. DETERMINING LOCATIONS FOR REPLICA SERVER PLACEMENT

In this step, given a certain number of replica servers to be placed, we need to decide the node locations for these servers. For this, we develop a heuristic algorithm called the *Server List Growing (SLG)* algorithm whose pseudocode is given as follows.

The algorithm chooses a node and then adds it to server list S that maximizes amount of content request load (hit rate) after evaluating every node that is not in S. Once a node is included, we will repeat the same process for the

Algorithm 1 Server List Growing (SLG) Algorithm

```
Input: A network topology $G$ with limited capacity on each link, the set of user nodes $U$ attached to each network node, the number of replica servers to be placed $A$, and the processing capacity and storage size at each replica server
Output: The node locations for replica server placement, i.e., $R_v$
1 Assuming that the origin server $s_o$ has been fixed by the content provider, we directly add the origin server to the server node list $S = \{s_o\}$.
2 Decide the content items to be cached at each node if this node is chosen for replica server placement based on the local content popularities at this node, i.e., run the UVP-based content caching algorithm (i.e., Algorithm 2) for each node (described in the next section).
3 For each network node $s$ in $G$ which is not in $S$, do
4     Run the content request load assignment algorithm (i.e., Algorithm 3) (described in the next section) to find, if $s$ is a replica server location, the maximum amount of user content request load can be served, denoted as $\theta_{S\bigcup\{s\}}$.
5 End for
6 After trying all the potential new replica server nodes $s$, choose node $s^* = arg \max_{s} \theta_{S\bigcup\{s\}}$ as the server node to be placed. Then, add the node $s^*$ to the server node list $S$, i.e., $S := S \cup \{s^*\}$.
7 Repeat steps from 3 to 7 until $A$ replica servers are placed.
```

remaining nodes until the total number of replica servers placed reaches A. This is a process where the server node list grows gradually. We have therefore called it the **Server List Growing (SLG) algorithm**. As a joint optimization effort, this algorithm incorporates the step of choosing content items to be cached (i.e., Algorithm 2) and the step of content request load assignment (i.e., Algorithm 3) in an integrated manner. This is expected to find a good solution and achieve better performance than other algorithms which do not do this jointly.

B. CHOOSING CACHED CONTENT FOR REPLICA SERVERS

We develop a UVP-based content caching algorithm given as Algorithm 2. Note that this step is relatively independent and can be implemented even before Algorithm 1. In fact, Step 2 in Algorithm 1 just implements this algorithm for each of the network nodes if it is chosen for replica server placement.

We order the content items by their popularity using the UVPs of users locally attached to the replica server. Then, according to the local popularity of each content item, we cache the most popular ones in the replica server. By caching popular content items close to users, we expect to be able to reduce the content delivery latency and

Algorithm 2 UVP-Based Algorithm

```
Input: The locations of replica servers, each user's UVP for each content item, i.e., $p^u_y$, which is obtained based on the assumption of the Zipf distribution
Output: The content items to be cached in each replica server, i.e., $\beta^y_v$
1 For each replica server $s$, do
2     For each content item $y$, sum up its UVPs of the users that are associated with the local server $s$ to find its popularity at the replica server $s$, i.e., $P^s_y = \sum_{u\in \boldsymbol{U_s}} p^u_y$, where $\boldsymbol{U_s}$ is the set of users that are locally attached to the replica server $s$.
3     Order the content items in a list $C_s$ according to $P^s_y$ from the largest to the smallest.
4     Get content items from $C_s$ one by one and cache them in the replica server $s$ until its maximum storage size is reached.
5 End for
```

improve significantly the network's overall link bandwidth usage.

C. ASSIGNING CONTENT REQUEST LOAD TO CONTENT SERVERS

With the replica server placement and the content items cached in the servers decided by the two earlier steps, we need to serve user content requests, where the anycast technique is assumed, i.e., the closest server that hosts the requested content and has sufficient remaining processing capacity would deliver the content to the requesting user. In order to maximize the amount of served content request load as well as to minimize both the content delivery latency and the network bandwidth consumption, we develop a heuristic algorithm called the *Closest First (CF)* algorithm, whose major idea is, with a server functioning as a central role, to use it to serve all its closest users. The pseudocode of the CF algorithm is given below.

In this algorithm, according to the remaining processing capacity of each server, we find all the closest users that have the requesting content items hosted by the server, and then use the remaining processing capacity of the server to serve these content request loads. There is a while loop inside the algorithm, which ensures that after one iteration of fully utilizing the remaining processing capacity of the servers, one more iteration is repeated. The benefit of this is that we can ensure that all the servers are serving the closest content request loads from users. In the content serving process, each server plays a central role to find the closest users, which is why we call this the *Server-based Closest First (CF)* algorithm.

D. COMPUTATIONAL COMPLEXITY ANALYSES

The computational complexity of the heuristic algorithm is analyzed as follows. In Algorithm 3, the steps within the while loop have the computational complexity of

Algorithm 3 Server-Based Closest First (CF) Algorithm

Input: The locations of the servers with their respective processing and storage capacity given, the contents cached in each server, and the network topology G with the associated link capacities.

Output: (1) The total amount of served content request load, i.e., $\sum_{u\in \boldsymbol{U},v\in \boldsymbol{N},y\in \boldsymbol{C}} \theta_y^{u,v}$, (2) the amount of content request load served by each server for each user, i.e., $\theta_y^{u,v}$, and (3) the total weighted content delivery latency, i.e., $\sum_{u\in \boldsymbol{U},v\in \boldsymbol{N},y\in \boldsymbol{C},i\in \boldsymbol{N}\cup \boldsymbol{U},j\in \boldsymbol{N_i}} \left(t_{ij}\cdot\theta_{i,j,y}^{u,v}\right) + \sum_{u\in \boldsymbol{U},v\in \boldsymbol{N},y\in \boldsymbol{C}} (\theta_y^{u,v}\cdot Q)$

```
1   Initialize the processing capacity of each server and
    the capacity of each network link; their capacities are
    normalized to one unit for delivering each content.
2   While ($P.size()>0\&\&R.size()>0$), where $P$ is the
    server list including the servers that still have remaining
    processing capacity and $R$ is the user list including
    all the users whose content request loads are not fully
    served.
3       For each content $y \in C$, do
4           Choose all the servers from the list $P$ that
            cache content item $y$, and put them in the
            server list $S_y$.
5           For each user $u \in R$, do
6               If the content request load for content $y$
                by user $u$ is not served, i.e., $f_y^u > 0$
7                   Choose the closest server $s$ from server
                    list $S_y$, and add the user $u$ to the user
                    list $U_s$ of server $s$
8               End if
9           End for
10          For each server $s \in S_y$, do
11              Sort the users in the list $U_s$ from the closest
                to the farthest
12              For each user $u \in U_s$, do
13                  Use the remaining processing capacity
                    of server $s$ to serve the content request
                    load of user $u$
14                  Record the content request load of user
                    $u$ served by server $s$, i.e., $\theta_y^{u,s}$, and
                    calculate the related load-weighted
                    content delivery latency
15                  Update the remaining content request
                    load of user $u$, the remaining processing
                    capacity at server $s$, and the remaining
                    capacity on each link. Remove the
                    user from user list $R$ if content request
                    has been completed, and remove the
                    server from the list $P$, if its processing
                    capacity has been used up.
16              End for
17          End for
18      End for
19  End while
20  Calculate the total amount of content request load
    served and the total load-weighted content delivery
    latency according to the first and second terms in (1).
```

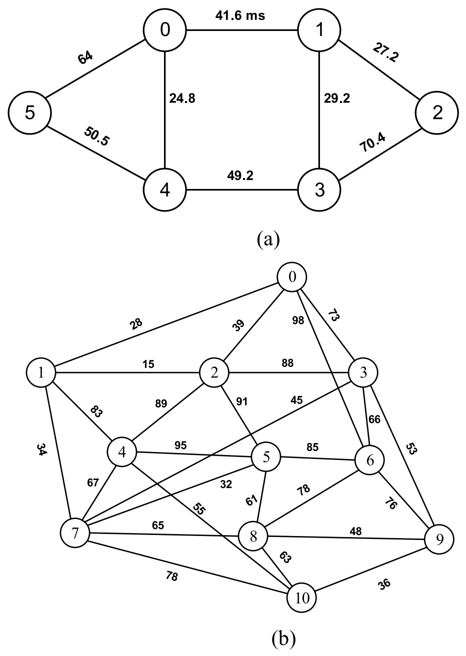


FIGURE 3. Test networks. (a) One representative 6-node, 8-link n6s8 network. (b) 11-node, 26-link COST239 network.

$O\left((|\boldsymbol{C}|+|\boldsymbol{A}|)\cdot|\boldsymbol{U}|\right)$, where $|\boldsymbol{C}|$ is the total number of content items to be cached,$|\boldsymbol{A}|$ is the number of replica servers to be placed in the network, and $|\boldsymbol{U}|$ is the total number of users in the network. Algorithm 2, which decides the contents to be cached in the replica servers, has the computational complexity of $O\left(|\boldsymbol{N}|\cdot|\boldsymbol{U}|\cdot|\boldsymbol{C}|\right)$, where $|\boldsymbol{N}|$ is the total number of nodes in the network. Note that, Algorithm 1 actually calls Algorithms 2 and 3 in its subroutines. (More specifically, Step 2 calls Algorithm 2 and Step 4 calls Algorithm 3.) Thus, the SLG algorithm is an integrated algorithm to jointly incorporate the effort of replica server placement, content caching, and content request load assignment. Algorithm 1 has the computational complexity of $O\left(|\boldsymbol{A}|\cdot\left(O\left(Alg.2\right)+|\boldsymbol{N}|\cdot O\left(Alg.3\right)+|\boldsymbol{N}|\cdot|\boldsymbol{U}|\right)\right)$, where $O\left(Alg.2\right)$ is the computational complexity of Step 2, $O\left(Alg.3\right)$ is for Step 4, and $|\boldsymbol{N}|\cdot|\boldsymbol{U}|$ is for Step 6.

VI. TEST CONDITIONS AND RESULTS

A. TEST CONDITIONS

We evaluated the performance of our replica server placement and content caching strategy by running simulations for the two kinds of test networks. These are (a) ten different topologies of six-node, eight-link n6s8 network, and (b) the 11-node, 26-link COST239 network. Fig. 3(a) shows one representative topology of the ten n6s8 networks, and the topology of COST239 network is shown in Fig. 3 (b). The value close to each link shows its transmission delay t_{ij} in units of ms. Note that the transmission delay between a server and a user that is locally connected is assumed to be 1 ms. A single origin server is assumed here though it is also possible to consider multiple origin servers. In both the test networks, we assumed that node 4 is placed with an origin server. The origin server is assumed to host all the contents and has a much higher processing capacity than the replica servers. The location of the origin server is generally predetermined by the content provider. The content users are randomly distributed in the network with different switching nodes hosting different numbers of users. The popularity of each content item for each user is assumed to follow the Zipf distribution according to the equation $\sum_{i=1}^{v} \frac{W}{i^a} = 1$. As mentioned earlier, this popularity is used to measure the normalized content request load of this content item from this user. The required storage size of each content item is assumed to be uniformly distributed within the range of [200, 400] units.

We employed the commercial AMPL/Gurobi [34] software package (version 6.5.1) to solve the MILP model on a 64-bit server with 2.4 GHz CPU and 8 GB memory. The MIPGAP of the MILP model is set to be 0.01%. We employed Java to implement the heuristic algorithms.

We evaluated the performance of the proposed approaches for replica server placement and content caching at the servers in terms of the maximum served content request load and the average content delivery latency. The average content delivery latency is calculated as the total load-weighted content delivery latency divided by the total content request load in the whole network.

For performance comparison, the SLG algorithm was employed for server placement. For content caching, in addition to the proposed UVP-based strategy, we also considered a *random caching strategy*, which randomly chooses the content items to be cached in the replica server until the storage capacity of the server is exhausted. Finally, for content request load assignment, in addition to the server-based CF algorithm, we also considered a simple *user-based closest first (CF)* assignment scheme. In this CF-assignment scheme, each user plays a central role, assigning the content request load of each user to its closest servers until either the user's content request load is fully served or there are no eligible replica servers left with remaining capacity to serve the load. After one user's content request load is processed, this CF-assignment scheme then moves to the next user until all the users' content request loads are processed. The overall process is a type of user-based best effort approach, which tries to satisfy the content request load of each user all at once.

B. IMPACT OF NUMBER OF REPLICA SERVERS PLACED

We first evaluate the performance impact of the number of replica servers placed for content delivery. Figs. 4(a) and (b) show the simulation results for ten n6s8 networks in terms of the ratio of unserved content request load and the average content delivery latency with an increasing number of replica servers placed in the network. Each data value presented in Fig. 4(a) and Fig. 4 (b) is the average of the results for the ten n6s8 networks. Note that the number of servers on the x-axis also includes the origin server. As a case study, we assume that the maximum normalized processing capacity of each replica server is 10 units, the processing capacity of the origin server is 30 units, and the maximum storage capacity in each replica server is 1000 units. The transmission capacity of each network link is assumed to be 10 units. We also assume that 10 content items are requested in the network and that 60 users are assumed to request these contents to generate the content request load following the *Zipf Distribution*. The distribution of these users between the network nodes is assumed to be as given in Table I.

TABLE 1. Number of local users of each network node in n6s8.

Node ID	N0	N1	N2	N3	N4	N5
# of users	7	12	7	13	6	15

For showing the performance results, the legend "SLG" corresponds to the SLG algorithm. The legends "UVP" and "Random" correspond to the UVP-based and random content caching strategies, respectively. The legends "SerBased" and "UserBased" correspond to the server-based and user-based CF algorithms for user content request load assignment, respectively. In addition, the legend "MILP" corresponds to the results obtained by the MILP optimization model for joint replica server placement, content caching, and content request load assignment.

As would be expected, when the number of replica servers deployed in the network increases, both the ratio of unserved content request load and the average content delivery latency decrease. This is reasonable because as the number of replica servers increases, more request load can be served. Moreover, when there are more replica servers deployed in the network we would expect to have servers closer to the end users. This would also tend to reduce the average content delivery latency.

For the ten n6s8 networks, the results in Figs. 4(a) and (b) show that the SLG_UVP_SerBased scheme performed the best to show the lowest ratio of unserved content request load and the shortest average content delivery latency among all the heuristic schemes. This scheme performs very close to the MILP model with only marginally poorer performance. In addition, comparing the results of the different caching strategies, we see that between them, the UVP-based scheme outperforms the random caching strategy by caching more of the popular contents at the replica servers.

Comparing the results of the two content request load algorithms, i.e., server-based CF and user-based CF, we see that the server-based scheme performs somewhat better than the user-based scheme in terms of the ratio of unserved content request load. The server-based CF also shows a clear performance advantage over user-based CF terms of the average delivery latency achieved. This happens because the server-based scheme always tries to serve users that are closest to the servers, thereby minimizing the network capacity used.

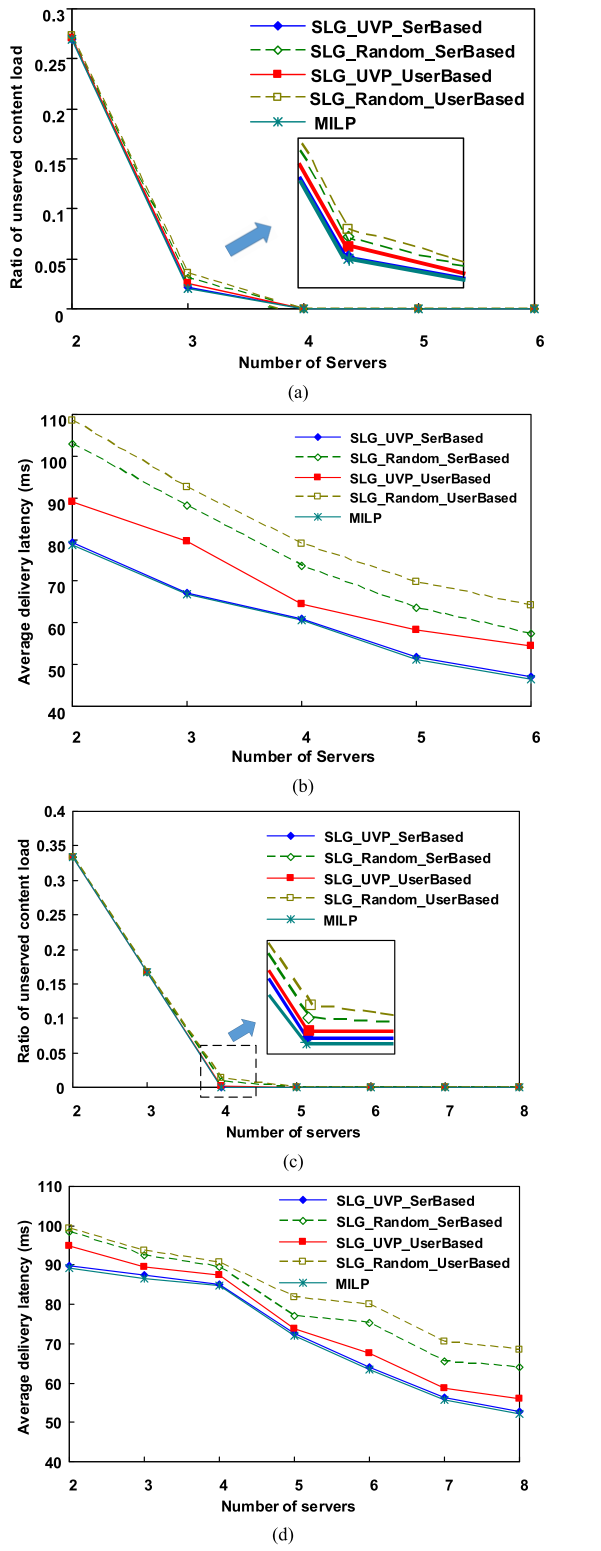


FIGURE 4. Performance change with an increasing number of replica servers. (a) Average ratio of unserved content request load (n6s8). (b) Average content delivery latency (n6s8). (c) Ratio of unserved content request load (COST239). (d) Average content delivery latency (COST239).

In contrast, the user-based scheme uses the closest servers to serve its content request load even though the closest servers may include several that are actually far away. If that happens then even though the entire content request load is served, much more link bandwidth is consumed and the average delivery latency may be more than that of the server-based scheme. In addition, the user-based scheme may also be intrinsically unfair. This is because the users that are assigned resources earlier would have a better chance to get served with a higher quality than those assigned with resources later; the latter may not only get a lower quality of service but also may not even be fully served.

We also made a similar performance comparison between the different schemes for the larger COST239 test network, where the system parameters are the same as those for the n6s8 network except that the link capacity is assumed to be 7 units (instead of the 10 units used earlier). As before, the 60 users in the network are randomly distributed and are assumed to request the contents randomly following the *Zipf Distribution*.

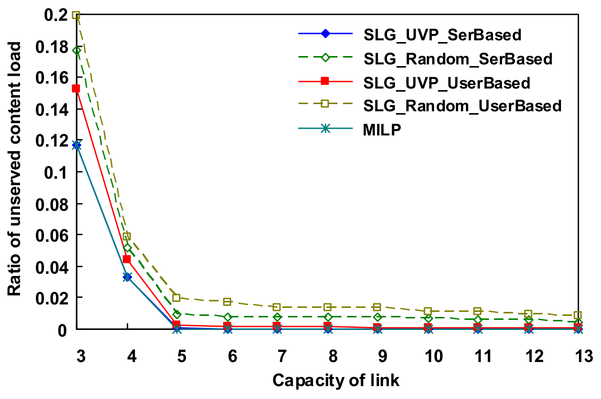
Figs. 4(c) and (d) show the results for the COST239 network. The performance trends between the different schemes for the COST239 network are similar to that observed earlier for the n6s8 network. The SLG-based replica server placement with UVP-based caching strategy and with server-based CF content request load assignment achieves the best performance between all the heuristic schemes and actually performs very close to the MILP model. Again, the UVP-based caching strategy can achieve better performance than the random caching strategy, and the server-based load assignment is more efficient than the user-based approach.

Figs. 4(a)-(d) show how increasing the actual number of suitably placed replica servers available in the network affects performance of the CDN. It is also important to note that the overall network performance is also significantly affected by several other important network parameters such as network link capacity, the replica server processing capacity, and the replica server storage capacity. Results illustrating the impact of these parameters on the overall performance of the CDN are given next.

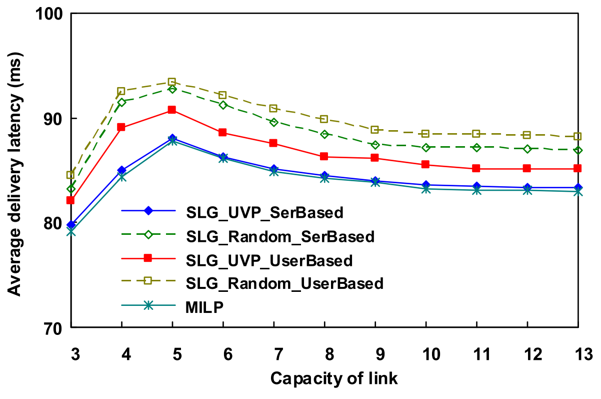
C. IMPACT OF NETWORK LINK CAPACITY

The impact of link capacity on the CDN performance is shown in Fig. 5. This shows the ratio of unserved content request load and average content delivery latency as a function of the network link capacity. In the interests of space, we have only presented here the results for the COST239 network as essentially similar observations can also be made for the performance of the smaller n6s8 network. Here, the link capacity is normalized with one unit delivering one unit of content. We assume that there are a total of 3 replica servers placed in the network and that all the servers have the same processing capacity as in the example of Fig. 4.

As observed earlier in Fig. 4, we find that the SLG scheme with UVP-based cached strategy and served-based load assignment algorithm is the most efficient with performance very close to that of the optimized MILP model. The UVP-based caching strategy is more efficient than the random strategy and the server-based load assignment effort



(a)



(b)

FIGURE 5. Performance change with increasing link capacity. (a) Ratio of unserved content request load (COST239). (b) Average content delivery latency (COST239).

achieves better performance than the user-based scheme. This trend is also observed when we evaluate the impact of the other parameters, i.e., the replica server processing capacity and storage size, in the subsequent sections. In the interests of space, we have not repeated discussion of this comparison in the following sections.

In Fig. 5(a), we see that with increasing network link capacity, the performance in terms of the ratio of unserved content request load improves, i.e., decreases. This is because there is now more bandwidth for content delivery from the remote content servers in case a local server does not have the content requested by a user in its cache. We also see a saturation trend here indicating that as far as the ratio of unserved load is concerned, increasing the link capacity beyond a threshold value results in only a marginal decrease in the unserved ratio. This is expected, because once the link capacity is high enough, it ceases to be the bottleneck limiting the success of the system in serving the contents requested.

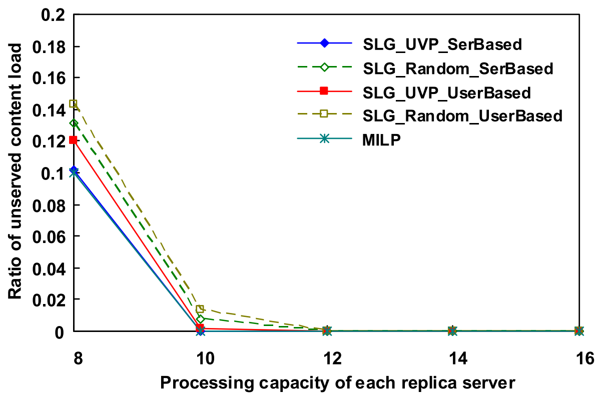
Fig. 5(b) shows the average content delivery latency as a function of the link capacity. One again, we observe the same performance trends between the different schemes as those in Fig. 5(a). However, it is interesting to note that the average content delivery latency shows a maxima as a function of the link capacity and that eventually, it also becomes saturated as before with increasing capacity. We do expect the maxima to occur, as even though more user content requests can be met with increasing link capacity, these may need progressively longer transmission paths. In turn, this would then tend to increase the delivery latency observed in the system. In Fig. 5(b), we observe that when going from 3 units of link capacity to 5 units, the ratio of unserved content request load decreases significantly (actually reaches zero) but that the average delivery latency increases. After 5 units of link capacity, a further increase of link capacity would not result in more content requests being served, but would help to reduce the average content delivery latency as there is now a greater likelihood for the content request loads to be served by closer servers. However, this reduction in the delivery latency eventually saturates when the link capacity does not put any limit on the successful ratio for content delivery (but the processing capacity of those servers might still have an effect). Thus, we can see that from 5 units of link capacity to 13 units, the average delivery latency reduces until it eventually reaches a saturation value.

D. IMPACT OF REPLICA SERVER PROCESSING CAPACITY

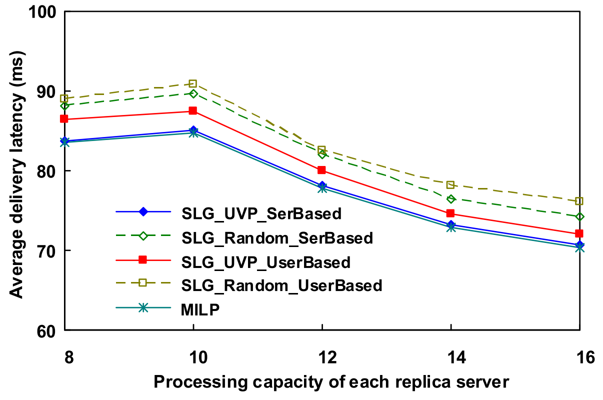
To study the impact of the processing capacity of the replica servers, we consider its impact on the ratio of unserved content request load and the average delivery latency. These are shown in Fig. 6 for the COST239 network, as functions of the total processing capacity of each replica server. (Once again, even though we did obtain these results for the n6s8 network, these were essentially similar and have not been shown here in the interests of space.) In Fig. 6, the processing capacity is normalized to one unit processing for each unit of content load. We assume that there are a total of 3 replica servers placed in the network and that each link has 7 units of normalized capacity.

Since the server is able to work faster when its processing capacity is higher, the number of served requests naturally increases with increasing server processing capacity, and therefore the ratio of unserved content request load decreases as shown in Fig. 6(a). As before, we also see a saturation trend indicating that beyond a threshold, increasing the server processing capacity further improves the ratio of unserved content request load only marginally. This is also expected as once the servers have enough processing capacity to process all the user request loads, this is no longer a bottleneck in the system and increasing this further is not likely to decrease the ratio of unserved content request load. This leads to the saturation trend observed in Fig. 6.

For the average content delivery latency in Fig. 6(b), as the server processing capacity increases, the overall average delivery latency tends to decrease. This is reasonable since a higher processing capacity in each server implies that more requests can be served locally, which would reduce the content delivery latency. However, when the processing capacity is high enough, most requests will be served locally with low latency. In that case, increasing the server processing capacity will once again not be significantly beneficial and a saturation trend is once again observed.

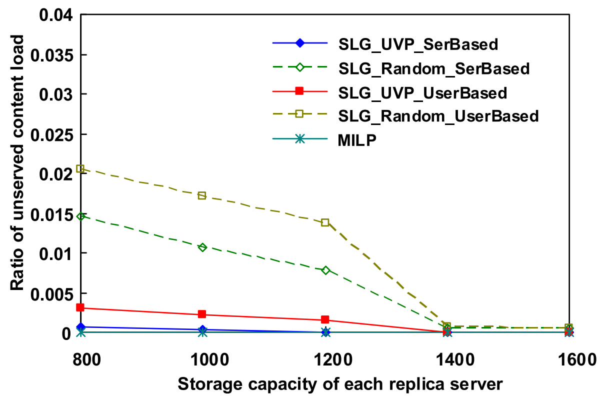


(a)

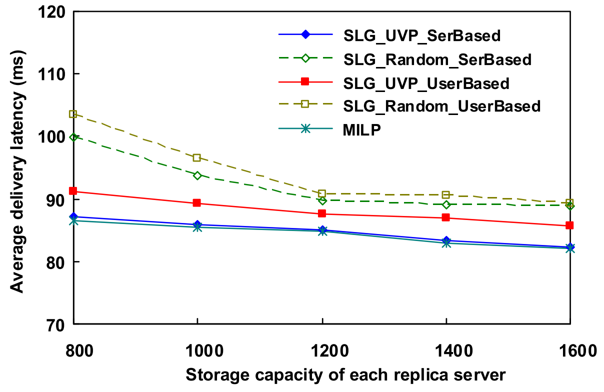


(b)

FIGURE 6. Performance change with increasing server processing capacity. (a) Ratio of unserved content request load (COST239). (b) Average content delivery latency (COST239).



(a)



(b)

FIGURE 7. Performance change with increasing server storage capacity. (a) Ratio of unserved content request load (COST239). (b) Average content delivery latency (COST239).

In Fig. 6(b), it is also interesting to see that when going from 8 units of processing capacity to 10 units, there is a minor increase in the average delivery latency. This is attributed to the fact that, in this case, a portion of content request load that will now be served by remote servers when the processing capacity of the local server is small and gets exhausted.

E. IMPACT OF REPLICA SERVER STORAGE CAPACITY

A similar study was conducted on how the replica server storage capacity would affect the performance of the system. These results are shown in Fig. 7 where we assume that there are a total of 3 replica servers placed in the network. As before, we assume that the processing capacity of each replica server is 10 units, and each link has 7 units of normalized capacity. Here, in the interests of space, we have once again only shown the results for the COST239 network as the n6s8 network behaves in a similar fashion.

A replica server with a larger storage capacity will be able to cache more contents and will therefore be able to satisfy more user requests locally. Therefore, the ratio of unserved content load improves with larger storage capacity as observed in Fig. 7(a). As before, the saturation trend observed is because of the fact that once the storage is enough to cache most of the popular contents, further increase in storage capacity is unlikely to give significantly more benefits. Instead, the server processing capacity at each server is then likely to become the bottleneck in serving user requests. Moreover, since more user requests can be served by the local servers as the replica server storage size increases, a lower average content delivery latency can be expected. This is indeed observed in Fig. 7(b).

VII. CONCLUSION

We focused on the design of efficient CDNs to maximize the hit ratio and minimize the average delivery latency of contents. For this, we developed an MILP model to jointly optimize replica server placement, content caching, and content request load assignment. We have also proposed efficient heuristic approaches for this optimization problem. Through simulation studies, we evaluated the performance of these approaches in terms of the ratio of unserved content request load and the average content delivery latency. We showed that the proposed SLG scheme with the UVP-based content caching and the server-based CF load assignment algorithm is much better than other approaches and is efficient enough to perform very close to the MILP model. In addition, we also evaluated how the performance of CDN changes with different numbers of placed replica servers, link capacity, server processing capacity, and server storage capacity. It was observed that these aspects jointly affect the performance of a CDN. Several saturation trends were also observed on the performance to show that the performance improvement becomes very marginal once sufficient resources or a sufficient number of replica servers are provided.

ACKNOWLEDGMENT

This paper was presented in part at the ACP 2015 [35] and ACP 2016 [36].

REFERENCES

[1] V. Stocker, G. Smaragdakis, W. Lehr, and S. Bauer, "The growing complexity of content delivery networks: Challenges and implications for the Internet ecosystem," *Telecommun. Policy*, vol. 41, pp. 1003–1016, Mar. 2017. [Online]. Available: http://dx.doi.org/10.1016/j.telpol.2017.02.004

[2] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Comput. Netw.*, vol. 116, pp. 79–95, Apr. 2017.

[3] *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*, Cisco, San Jose, CA, USA, Jun. 2017.

[4] M. Dehghan *et al.*, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1635–1648, Jun. 2017.

[5] J. Sahoo *et al.*, "A survey on replica server placement algorithms for content delivery networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1002–1026, 2nd Quart., 2017.

[6] D. Sarkar, N. Rakesh, and K. K. Mishra, "Content delivery networks: Insights and recent advancement," in *Proc. 4th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Waknaghat, India, 2016, pp. 1–5.

[7] J. Perelló, K. Walkowiak, M. Klinkowski, S. Spadaro, and D. Careglio, "Joint content placement and lightpath routing and spectrum assignment in CDNs over elastic optical network scenarios," *Comput. Commun.*, vol. 77, pp. 72–84, Mar. 2016.

[8] J. Sahoo and R. Glitho, "Greedy heuristic for replica server placement in cloud based content delivery networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Messina, Italy, Jun. 2016, pp. 302–309.

[9] G. Sun, V. Chang, G. Yang, and D. Liao, "The cost-efficient deployment of replica servers in virtual content distribution networks for data fusion," *Inf. Sci.*, vol. 432, pp. 495–515, Mar. 2017. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2017.08.021

[10] H. Yin *et al.*, "Edge provisioning with flexible server placement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1031–1045, Apr. 2017.

[11] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. INFOCOM*, 2010, pp. 1–9.

[12] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity based and collaborative in-network caching for content-oriented networks," in *Proc. INFOCOM Workshop NOMEN*, 2012, pp. 316–321.

[13] Y. Kim and I. Yeom, "Performance analysis of in-network caching for content-centric networking," *Comput. Netw.*, vol. 57, no. 13, pp. 2465–2482, 2013.

[14] Z. Zhao, M. Peng, Z. Ding, W. Wang, and H. V. Poor, "Cluster content caching: An energy-efficient approach to improve quality of service in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1207–1221, May 2016.

[15] K. Kanai *et al.*, "Proactive content caching for mobile video utilizing transportation systems and evaluation through field experiments," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2102–2114, Aug. 2016.

[16] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.

[17] D. Malak, M. Al-Shalash, and J. G. Andrews, "Optimizing content caching to maximize the density of successful receptions in device-to-device networking," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4365–4380, Oct. 2016.

[18] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Joint smart pricing and proactive content caching for mobile services," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2357–2371, Aug. 2016.

[19] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1076–1089, May 2017.

[20] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching for low end-to-end latency in cloud-based wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[21] M. A. Salahuddin, J. Sahoo, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on content placement algorithms for cloud-based content delivery networks," *IEEE Access*, vol. 6, pp. 91–114, Feb. 2018.

[22] J. Sung, M. Kim, K. Lim, and J.-K. K. Rhee, "Efficient cache placement strategy in two-tier wireless content delivery network," *IEEE Trans. Multimedia*, vol. 18, no. 6, pp. 1163–1174, Jun. 2016.

[23] J. Sung, K. Kim, J. Kim, and J.-K. K. Rhee, "Efficient content replacement in wireless content delivery network with cooperative caching," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 547–552.

[24] T. A. Neves, L. M. A. Drummond, L. S. Ochi, C. Albuquerque, and E. Uchoa, "Solving replica placement and request distribution in content distribution networks," *Electron. Notes Discrete Math.*, vol. 36, pp. 89–96, Aug. 2010.

[25] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, "On the optimization of storage capacity allocation for content distribution," *Comput. Netw.*, vol. 47, no. 3, pp. 409–428, Feb. 2005.

[26] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. J. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 356–365, Apr. 2004.

[27] T. Bektaş, J.-F. Cordeau, E. Erkut, and G. Laporte, "Exact algorithms for the joint object placement and request routing problem in content distribution networks," *Comput. Oper. Res.*, vol. 35, no. 12, pp. 3860–3884, Dec. 2008.

[28] T. Neves, L. S. Ochi, and C. Albuquerque, "A new hybrid heuristic for replica placement and request distribution in content distribution networks," *Optim. Lett.*, vol. 9, no. 4, pp. 677–692, Apr. 2015.

[29] A. A. Haghighi, S. S. Heydari, and S. Shahbazpanahi, "Dynamic QoS-aware resource assignment in cloud-based content-delivery network," *IEEE Access*, vol. 6, pp. 2298–2309, 2017.

[30] T. Bektas, O. Oguz, and I. Ouveysi, "Designing cost-effective content distribution networks," *Comput. Oper. Res.*, vol. 34, no. 8, pp. 2436–2449, Aug. 2007.

[31] K. Lim, Y. Bang, J. Sung, and J.-K. K. Rhee, "Joint optimization of cache server deployment and request routing with cooperative content replication," in *Proc. ICC*, 2014, pp. 1790–1795.

[32] J. Choi, A. Reaz, and B. Mukherjee, "A survey of user behavior in VoD service and bandwidth-saving multicast streaming schemes," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 156–169, 1st Quart., 2012.

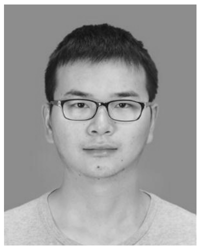
[33] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[34] AMPL & Gurobi. (2015). *Linear Programming Optimization Software Package*. [Online]. Available: http://www.gurobi.com

[35] X. Li and G. Shen, "Optimal content caching based on content popularity for content delivery networks," in *Proc. ACP*, 2015, pp. 1–3, paper AS4G.4.

[36] X. Li, X. Zhao, S. Bose, B. Chen, M. Gao, and G. Shen, "Optimal replica servers placement for content delivery networks," in *Proc. ACP*, 2016, pp. 1–3, paper AS2D.2.

KAI XU is currently with the School of Electronic and Information Engineering, Soochow University, China, and also with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, as a Ph.D. student. His research interest is in the area of broadband networks.



XIANG LI received the master's degree with the School of Electronic and Information Engineering, Soochow University, China. He is currently with Huawei Technology as a Research Engineer. His research interest is in the area of broadband networks.

SANJAY KUMAR BOSE (SM'91) received the B.Tech. degree from IIT Kanpur in 1976 and the master's and Ph.D. degrees from SUNY Stony Brook, USA, in 1977 and 1980, respectively. He was with the Corporate Research and Development Centre of the General Electric Company, Schenctady, NY, USA, until 1982. He joined IIT Kanpur as an Assistant Professor and became a Professor in 1991. He left IIT Kanpur in 2003 to join the Faculty of the School of EEE, NTU, Singapore. In 2008, he left NTU to join IIT Guwahati, where he is currently a Professor with the Department of EEE. He has concurrently held the position of Dean, Alumni Affairs, and External Relations in IIT Guwahati from 2011 to 2014.

He has been involved in various areas in the field of computer networks and queueing systems for the past three decades. He has published extensively in the areas of optical networks, network routing, modeling and analysis of networks and queueing systems and wireless networking. he has also authored various popular text books in the areas of queueing systems, microprocessor systems and hardware and software of personal computers. He hosts an extremely popular Web site on instructional material for queueing systems.

Prof. Bose is a fellow of IETE, India, and a member of Sigma Xi and Eta Kappa Nu.

GANGXIANG SHEN (S'98–M'99–SM'12) received the B.Eng. degree from Zhejiang University, China, the M.Sc. degree from Nanyang Technological University, Singapore, and the Ph.D. degree from the University of Alberta, Canada, in 2006. He was a Lead Engineer with Ciena, Linthicum, Maryland. He was also an Australian ARC Post-Doctoral Fellow with the University of Melbourne. He is currently a Distinguished Professor with the School of Electronic and Information Engineering, Soochow University, China. He has authored and co-authored over 150 peer-reviewed technical papers, among which one of the papers received the highest citations among all the papers published in the IEEE/OSA JOCN. His research interests include integrated optical and wireless networks, spectrum efficient optical networks, and green optical networks. He received the Young Researcher New Star Scientist Award in the 2010 Scopus Young Researcher Award Scheme, China. He was a recipient of the Izaak Walton Killam Memorial Award from the University of Alberta and the Canadian NSERC Industrial Research and Development Fellowship. He has served as TCP chairs for various international conferences in the area of optical networking, including the General TPC Co-Chair of ACP 2018 and the Symposium Lead Chair of GLOBECOM 2017. He was a Lead Guest Editor of the IEEE JSAC Special Issue on Next-Generation Spectrum-Efficient and Elastic Optical Transport Networks, and a Guest Editor of the IEEE JSAC Special Issue on Energy-Efficiency in Optical Networks. He is an Associate Editor of the IEEE/OSA JOCN and an Editorial Board Member of *Optical Switching and Networking* and *Photonic Network Communications*. He is a Highly Cited Chinese Research Scholar selected by Elsevier, from 2014 to 2017, and an Excellent Young Research Scholar sponsored by NSFC. He was a Secretary for the IEEE Fiber-Wireless Integration Sub-Technical Committee. He is serving as a member of IEEE ComSoc Strategic Planning Standing Committee and an IEEE ComSoc Distinguished Lecturer from 2018 to 2019.

• • •