Parametric-Decomposition Based Request Routing in Content Delivery Networks

Tuğçe Bilen[1(✉)], Dinçer Salih Kurnaz[2], Serkan Sevim[2], and Berk Canberk[1]

[1] Department of Computer Engineering, Faculty of Computer and Informatics, Istanbul Technical University, Ayazaga, 34469 Istanbul, Turkey
{bilent,canberk}@itu.edu.tr
[2] Medianova, Mecidiyeköy, 34387 Istanbul, Turkey
{dincersalih.kurnaz,serkan.sevim}@medianova.com

Abstract. Content Delivery Networks (CDNs) enable the rapid web service access by meeting the client requests using the optimal surrogate server located at their nearby. However, the optimal surrogate server can suddenly be overloaded by the spiky characteristics of the high-bandwidth client requests. This accumulates both the drop rates and response times of the client requests. To solve these problems and balance the load on surrogate servers, we propose a Parametric-Decomposition based request routing at the surrogate servers in CDNs. With the Parametric Decomposition method, we combine the high-bandwidth client requests on origin server with our proposed Superposition and Queuing procedures. Then, we split these requests into more than one surrogate server through proposed Splitting and Adjustment procedures. We model the origin and surrogate servers based on G/G/1 queuing system to determine the load status. In case of high congestion on the origin server, we split client requests to the different surrogate servers instead of selecting one. The split sizes of whole content are adjusted by defining a novel splitter index parameter based on the queuing load and waiting time of surrogate servers. The results reveal that the proposed strategy reduces the load on surrogate servers by 42% compared to the conventional approaches. Moreover, the latency and request drops are decreased by 44% and 57% compared to the conventional approaches, respectively.

Keywords: Content Delivery Networks · Load balancing
Request routing · Queuing theory

1 Introduction

With the rise of next generation cloud networking, the usage of CDNs enables higher throughput, accessibility, and bandwidth with smaller mean response time in internet core infrastructure [1]. Here, the high number of client requests to access web services may increase the congestion and bottleneck problems [2]. With the evolution of the CDNs, surrogate servers solve this congestion problem

K. R. Chowdhury et al. (Eds.): WWIC 2018, LNCS 10866, pp. 323–335, 2018.
https://doi.org/10.1007/978-3-030-02931-9_26

through content replication. In this way, client requests are intelligently routed to get original content from the nearest server.

Request routing is one of the most crucial functionality of CDNs. The main aim of the request routing is to direct the client requests to the nearest surrogate server. In this way, clients can reach the requested content with smaller response time and higher throughput. On the other hand, the nearest surrogate server may not be the best server in every situation. Accordingly, the load status, response, and waiting times of the surrogate servers must be taken into consideration during the routing procedure to select the optimal server.

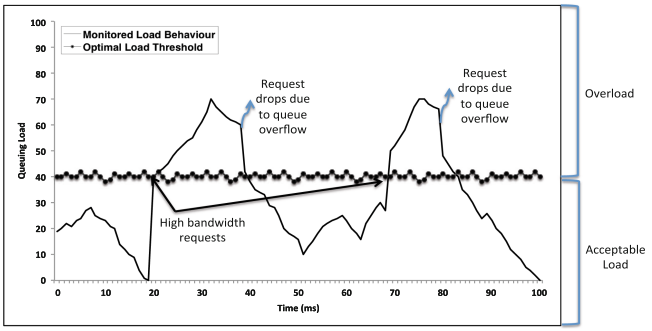


Fig. 1. Surrogate server overload caused by high-bandwidth requests

In some cases, clients can transfer many requests for high-bandwidth content. If these requests are routed to a specific server, the queuing load on that server is increased dramatically as shown in Fig. 1. This overload also increases the waiting times of the client requests in the queue. At this point, drop rate and response time of the clients begin to suddenly increase. These drops observed during the transfers make the web latency 5-times slower [3]. This situation causes a reduction in the client experience. For example, every 100ms increase in latency will reduce the profits by 1% for Amazon [4]. Therefore, the selection of optimal server without considering the request size decreases the performance of the CDN.

Thus, through this paper, we aim to investigate the effects of a high bandwidth client requests on the CDN request routing procedure. Also, we come up with a complete novel method called as Parametric-Decomposition to overcome this problem. The parametric-decomposition enable us to incorporate the high-bandwidth client requests on origin server and split again into more than one surrogate servers in a more autonomous way.

1.1 Related Work

As mentioned above, request routing distributes the client requests to the optimal surrogate servers to achieve load balancing. There are different request rout-

ing mechanisms in the literature as dynamic or static. Dynamic algorithms collect information from the network and servers to use in the routing decision. On the other hand, static algorithms cannot use any data gathering mechanism. As summarized in [5], the Random algorithm distributes the requests to the surrogate servers with uniform probability as a static approach. In the Round Robin, different surrogate servers are selected during each routing. Thus, the same number of the client request is transferred to each surrogate server statically. The Least-Loaded algorithm transfer the client requests to the least loaded server [6]. Similarly, the work in [7] executes the routing according to the available bandwidth and round-trip latency of the surrogate servers. Two Random Choices algorithm route the request to the least loaded server among the two choices [8]. The [9] proposes a network cost aware request routing to assign the user requests by reducing the server charging volume. In [10], the client requests are routed to other CDNs when the request for a video in a particular CDN could not be found. Also, the proximity and load on the servers are considered during these procedures. Similarly, [11] executes the CDN interconnection request routing with the help of Application-Layer Traffic Optimization. The article [12] solves the request mapping and routing problems at the same time with the distributed algorithm based on Gauss-Seidel to optimize the performance and cost. However, the routing of high-bandwidth client requests is not considered in none of these works.

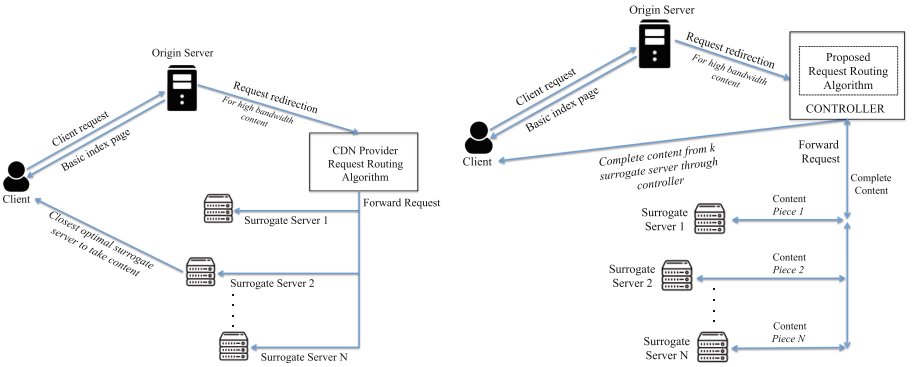
Contributions. In this paper, we propose a Parametric-Decomposition based request routing in CDNs. With the Parametric-Decomposition, the incoming client requests to the origin server can be combined and split again into the different surrogate servers. We aim to route the high-bandwidth client requests to more than one server with this strategy. In this way, we can reduce the load on a specific surrogate server causing from high-bandwidth requests. Also, we can reduce the latency and drop rates of the clients. The main contributions of the paper can be summarized as follows:

- We model the origin and surrogate servers according to the G/G/1 queuing system.
- In the modeling of the origin server, we define two procedures as Superposition and Queuing to determine the load status. We use load information of origin server in the content split decision.
- In the modeling of surrogate servers, we also define two procedures called as Splitting and Adjustment. The Splitting enables the queue load and waiting time parameters to the Adjustment procedure.
- In Adjustment procedure, the split content sizes are adjusted by a novel proposed splitter index parameter based on the queue load and waiting time of surrogate servers.

The rest of the paper is organized as follows: the proposed content delivery network architecture is given in Sect. 2. In Sect. 3, the request routing model is detailed. The performance of the proposed approach is evaluated in Sect. 4. Lastly, we conclude the paper in Sect. 5.

2 Content Delivery Network Architecture

In this paper, we use the partial-site approach as a basis for our CDN request routing method. In this approach, client requests are transferred to the origin server to return the main content. Then, the requests are sent to the optimal surrogate server to take the high-bandwidth contents as shown in Fig. 2a.



(a) Partial Site Request Routing Approach (b) Proposed Request Routing Approach

Fig. 2. CDN Request Routing Architectures for Traditional and Proposed Approaches

At this point, instead of selecting only one server, we split the client requests to different surrogate servers which are in the geographical proximity of the client as shown in Fig. 2b. For this aim, in this paper, we consider a CDN network architecture consisting of controller, clients, origin and surrogate servers as shown in Fig. 2b. The origin server contains the original data and distributes it to each of the surrogate servers accordingly. In this paper, we particularly use the cooperative push-based approach for content outsourcing which is based on the pre-fetching of content to surrogates [1]. The controller can communicate with the client, origin and surrogate servers to manage the proposed request routing approach as explained in the following section.

3 Request Routing Model

The proposed request routing approach is managed by the controller. For this purpose, the controller consists of the *Congestion Determination, Content Distribution, and Content Combination Modules* as shown in Fig. 3. The details of these modules are explained in following subsections.

3.1 Congestion Determination Module

In this module, first, the controller evaluates congestion level of the origin server to apply the proposed request routing approach. If the client sends a request

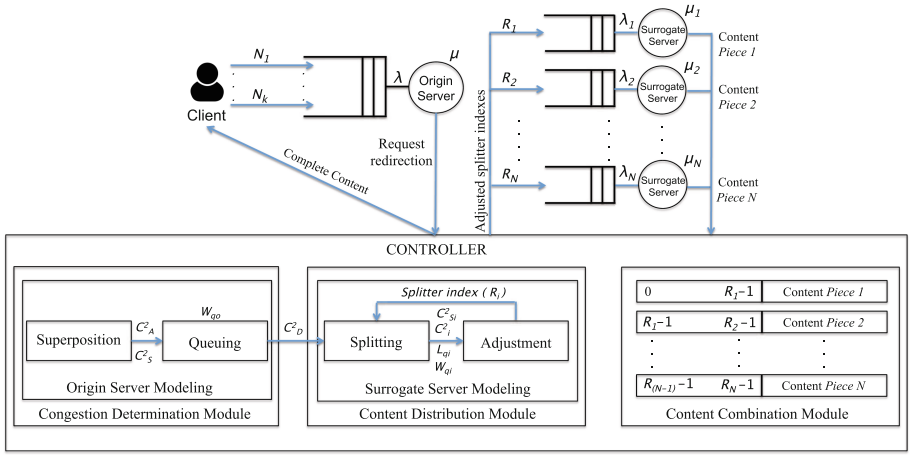


Fig. 3. Proposed system framework

for high-bandwidth content and the congestion level of the origin server rises above the optimal threshold, then we use our proposed request routing model. For this purpose, this module takes the client requests arriving at origin server as input. Respectively, it gives the congestion status information of the origin server. The congestion determination module also includes the *Origin Server Modeling* submodule with *Superposition* and *Queuing* procedures as explained in the following part.

Origin Server Modeling. In this paper, we use an approximation based on the two-parameter characterization of the arrival processes [13]. For the origin server two-parameter characterization, the first parameter λ is the mean arrival rate of the client requests to the origin server and it is equal to the $1/E[A]$. Here, A shows arrival time of the client requests to the origin server. The second parameter C_A^2 is the Squared Coefficient of Variation (SCV) of inter-arrival times. These two parameters λ and C_A^2 are related to the first and second moment of interarrival times. Correspondingly, we can observe the arrival characteristics of the client requests by using these parameters. For this purpose, we decompose the origin server modeling into two procedures as *Superposition* and *Queuing* as shown in Fig. 3.

Superposition: The client requests cannot reach to the origin server in a specific distribution. These requests come as random arrivals with high variability. Also, the total arrival stream of client requests to the origin server is the superposition of all requests transferred by the client. Therefore, we model the origin server according to the *G/G/1* queuing model. Appropriately, $N_j(t)$ is the number of transferred requests from the client to origin server by time t. The combined arrival process to origin server is equal to the $N(t) = \sum_{j=1}^{k} N_j(t)$, ($k$ is the maximum request number of client). At this point, the total ($N(t)$) and each of the

smaller streams ($N_j(t)$) are the renewal processes. By using these parameters, the arrival rate of client requests to the origin server is $\lambda_j \equiv \lim_{t\to\infty} \frac{E[N_j(t)]}{t}$. Consequently, the mean arrival rate $\lambda = \sum_{j=1}^{k} \lambda_j$ equals to the $\lim_{t\to\infty} \frac{E[N(t)]}{t}$. After finding the λ, we can reach the SCV of inter-arrival time to the origin server by using the Eq. 1.

$$C_A^2 = \frac{1}{\lambda} \sum_{j=1}^{k} \lambda_j C_j^2 \tag{1}$$

In the Eq. 1, C_j^2 represents the SCV between the departures from client to the origin server and $C_j^2 = \vartheta_i/\lambda_j$. Here, the ϑ_j is limiting factor and calculated as $\vartheta_j \equiv \lim_{t\to\infty} \frac{Var[N_j(t)]}{t}$. Furthermore, the service rate of the origin server is the $\mu = \frac{1}{E[S]}$ and here, S is the random service time of the origin server. Then, the SCV of service distribution at origin server is the $C_S^2 = \frac{Var[S]}{E[S]}$.

Queuing: The client requests first arrive at the origin server, receive basic content, and then transferred to the surrogate servers through the controller for taking the high-bandwidth contents. Hence, the SCV between departures (C_D^2) from the origin server is found as given in Eq. 2.

$$C_D^2 = \rho^2 C_S^2 + (1-\rho^2) C_A^2, \quad (\rho = \frac{\lambda}{\mu}) \tag{2}$$

According to the *G/G/1* queuing model, the approximation of queue waiting time (W_{q_o}) for the origin server is found as given in Eq. 3.

$$W_{q_o} = E(S) \frac{\rho}{1-\rho} \frac{C_A^2 + C_S^2}{2}, \quad (\rho = \frac{\lambda}{\mu}) \tag{3}$$

Also, by using Eq. 3, $I = \frac{W_{q_o}}{E(S)}$ is defined as the congestion index of the origin server. If the congestion index approaches to 1, then the origin server starts to be congested and this is an indicator of the heavy traffic (asymptotically exact as $\rho \uparrow 1$). Similarly, the values of C_A^2 and C_S^2 show variability level of the arrival and service distributions. The variability accumulates congestion, therefore, higher values of these parameters show congestion on the origin server. Therefore, in addition to the I, the controller uses C_A^2 and C_S^2 parameters to determine congestion status of the origin server. In the overload case, we split the client requests to different surrogate servers. At this point, the calculated C_D^2 value is transferred to content distribution module for use in the client request splitting procedure to the surrogate servers as explained in the following subsection.

3.2 Content Distribution Module

In this module, first client requests leaving from the origin server are split to the different surrogate servers in equal sizes through the controller. Then, the

split content sizes are determined with the *splitter index* parameter by considering the queue load and waiting time of the surrogate servers by the aid of SCV parameters. For these purposes, the content distribution module includes the *Surrogate Server Modeling* submodule with *Splitting* and *Optimization* procedures as explained in the following part.

Surrogate Server Modeling. We also use the *G/G/1* queuing system for modeling the surrogate servers. The arrival and service times of the surrogate servers are independent and identically distributed (IID). Accordingly, the arrival and service rates of the surrogate servers are $\lambda_i = \frac{1}{E[A_i]}$ and $\mu_i = \frac{1}{E[S_i]}$. Here, A_i and S_i represent the arrival and service times of the surrogate servers, respectively.

Splitting: The client requests for high-bandwidth contents are split to different surrogate servers by the controller. The SCV between these routed arrival requests to the specific surrogate server is calculated by using the Eq. 4.

$$C_i^2 = R_i C_D^2 + (1 - R_i), \quad \forall i = 1, ..., N \tag{4}$$

In this equation, C_D^2 is obtained from the Queuing procedure as given in Eq. 2. Also, we define a R parameter called as *splitter index* based on the queue load and waiting time of surrogate servers to determine the split piece sizes of whole content in Adjustment procedure. Subsequently, to determine the splitter index (R_i), we require the queue loads and waiting times of the surrogate servers. According to the *G/G/1* queuing model, the queue waiting time approximation of the surrogate server is found as given in Eq. 5.

$$W_{q_i} = E(S_i) \frac{\rho_i}{1 - \rho_i} \frac{C_i^2 + C_{Si}^2}{2}, \quad (\rho_i = \frac{\lambda_i}{\mu_i}) \tag{5}$$

In Eq. 5, $C_{Si}^2 = \frac{Var[S_i]}{E[S_i]}$ is the SCV of the service distribution at the surrogate server. Moreover, according to the Little's theorem, the queue loads of the surrogate servers are found as given in Eq. 6.

$$L_{q_i} = \frac{\rho_i^2}{1 - \rho_i} \frac{C_i^2 + C_{Si}^2}{2}, \quad (\rho_i = \frac{\lambda_i}{\mu_i}) \tag{6}$$

Adjustment: As explained above, the high-bandwidth client request is split into different surrogate servers and R_i is the *splitter index* to determine the optimal split size of the whole content to a specific surrogate server.

Initially, we take the *splitter index* parameters of surrogate servers are equal ($R_i = \frac{1}{N}$, $i = 1, 2, .., N$) and so, the content is divided into pieces of equal sizes ($\frac{K}{N}$, K: size of whole content). Then, to determine the optimal split sizes of content, we adjust the splitter index according to the queue loads and waiting

Table 1. Key notations

Notation	Explanation
λ, μ	Arrival and service rates of origin server
λ_i, μ_i	Arrival and service rates of surrogate servers
R_i	Splitter index
C_A^2, C_S^2	SCVs of inter-arrival time and service distribution at origin server
C_D^2	SCV between departures from origin server
C_j^2	SCV between departures from client to origin server
C_i^2, C_{Si}^2	SCVs of routed arrival requests and service distribution at surrogate server

times of surrogate servers by the aid of SCV parameters. Therefore, we define the splitter index as a function of the queue load and waiting time as given in Eq. 7.

$$R_i = \frac{\frac{L_{q_i}(C_i^2, C_{Si}^2)}{\sum_{i=1}^{N} L_{q_i}(C_i^2, C_{Si}^2)} + \frac{W_{q_i}(C_i^2, C_{Si}^2)}{\sum_{i=1}^{N} W_{q_i}(C_i^2, C_{Si}^2)}}{2}, \forall i = 1, 2, .., N \tag{7}$$

In the definition of R_i, the first parameter in sum is obtained from the *queue load based R_i adjustment*. After finding the $R_i = \frac{1}{N}$, we calculate the C_i^2 values of the corresponding surrogate servers with Eq. 4. Then, the calculated C_i^2 is used in Eq. 6 to find L_{q_i}. These procedures are executed for all surrogate servers and corresponding load values are collected to obtain a total load value. The second parameter in sum comes from the *queue waiting time based R_i adjustment*. Here, the same procedures are executed on the W_{q_i} parameter. Finally, we take the average of the load and queue waiting time-based adjustments to find the splitter index. Therefore, the splitter index (R_i) parameter of each surrogate server is found as given in Eq. 7. Finally, each splitter index (R_i) is multiplied by K to find the split size of whole content to that server. The procedures to adjust the split content size is also summarized in Algorithm 1. Also, the key notations used in all equations through the paper is given with Table 1.

3.3 Content Combination Module

As mentioned above, the controller transfers split content to different surrogate servers according to the calculated splitter index (R_i) parameters. Before transferring whole content to the client, the controller should combine split content pieces taken by these surrogate servers again. To ease the combination of content pieces, we define a sequence range in the packet header. Accordingly, the controller should add the sequence range to the header of each transferred content piece according to R_i parameters. Therefore, the controller combines content pieces according to these sequence ranges before transferring the client as given in Fig. 3.

Algorithm 1. Content Size Splitting

$L_T = 0$
$W_T = 0$
for $i \longleftarrow 1$ to N **do**
    Calculate $R_i = \frac{1}{N}$
end for
Queue Load Based Adjustment()
for $i \longleftarrow 1$ to N **do**
    Calculate C_i^2 with Eq. 4
    Calculate L_{q_i} with Eq. 6
    Add L_{q_i} to L_T
end for
Queue Waiting Time Based Adjustment()
for $i \longleftarrow 1$ to N **do**
    Calculate C_i^2 with Eq. 4
    Calculate W_{q_i} with Eq. 5
    Add W_{q_i} to W_T
end for
for $i \longleftarrow 1$ to N **do**
    $R_i = \frac{L_{q_i} + W_{q_i}}{2(L_T + W_T)}$
    Assign KR_i to surrogate server i
end for

4 Performance Evaluation

4.1 Simulation Setup and Methodology

The efficiency of the proposed approach is evaluated through simulations by comparing with Round Robin, Least-Loaded, and Two Random Choices request routing algorithms. The simulations are executed by using the ns2 network simulator. To the best of our knowledge, the current simulators do not include any tool for evaluating the different request routing mechanisms. Accordingly, we benefit from the new library added to the ns2 network simulator during the implementation of our request routing approach [14]. Also, we use network topology of the Medianova Company. Each node on this network corresponds to a surrogate server which connects to the end users and an origin server. The metrics used to evaluate performance are queue load, the latency of client, and request drops due to queue overflow.

4.2 Performance Results

Queue Load. We first evaluate the queue load behavior of a specific surrogate server according to the time. The queue length of this server in time help us to observe the load behavior and we use the Eq. 6 for this purpose. Here, requests

from the client are transferred to this surrogate server in Round Robin, Least-Loaded, and Two Random Choices request routing algorithms. In our approach, we use other servers in addition to this server during the high-bandwidth client requests.

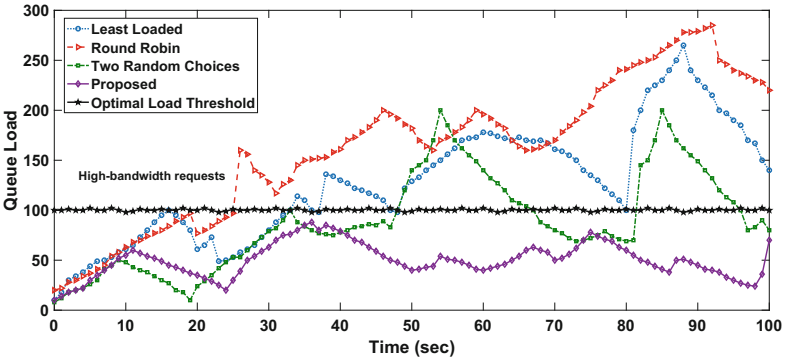


Fig. 4. Queue load behavior of a server

The Round Robin does not collect information about the network and server status due to it is a static algorithm. Although the Least-Loaded and Two Random Choices algorithms are dynamic, they do not consider the routing of high-bandwidth client requests. Accordingly, as shown in Fig. 4, the transferred two different high-bandwidth requests cause overloading of the server in these three algorithms. On the other hand, we use other servers to meet the client requests based on Algorithm 1. Accordingly, as shown in Fig. 4, we have roughly 42% less load on this specific server compared to other three algorithms.

Latency of Client. We also evaluate the latency of client until the request is met. We define this latency as the difference between the sending time of request by the client and its response time by the surrogate server. During this duration, client requests are kept in the queues of the origin and surrogate servers. Therefore, we use the waiting times of the client requests in these queues.

Accordingly, we use the Eqs. 3 and 5 for the origin and surrogate server waiting times calculations, respectively. Additionally, the load status of the surrogate servers affects the latency of the requested client. The increased number of client accumulates the requests waiting in queues of servers. Therefore, we investigate the latency according to the client number parameter. As shown in Fig. 5, in the Round Robin, Least-Loaded, and Two Random Choices algorithms, the latency of a client increases with growing request number. More specifically, client observes more latency during the high-bandwidth request transfers. In these durations, high-bandwidth requests of client and other increased requests accumulates the latency and drops. On the other hand, we do not overwhelm the

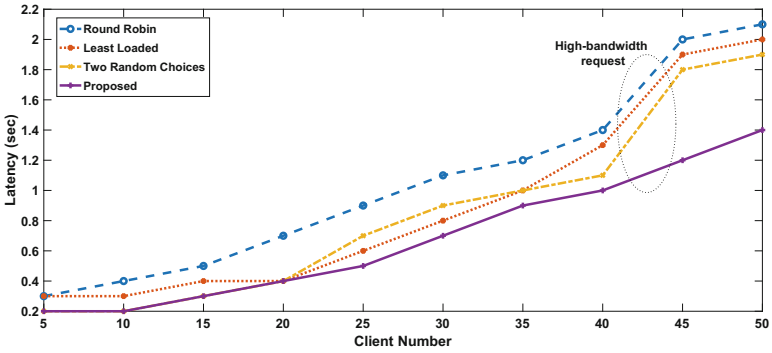


Fig. 5. Latency of a client with increased request numbers

specific server with high-bandwidth requests in a very demanding environment based on Algorithm 1. Therefore, as shown in Fig. 5, we have roughly 44% less client latency compared to other three algorithms.

Request Drops Due to Queue Overload. To show that our proposed approach does not overload the queues of servers, we investigate the dropped requests of clients by the surrogate servers. We observe the request drops again the client number parameter. As explained above, loads of surrogate server queues are accumulated with increasing client and request numbers. This load on the surrogate servers increases the request drops of clients. For this reason, as shown in Fig. 6, we observe an increase in the drops during the incoming of high-bandwidth requests for the Round Robin, Least Loaded, and Two Random Choices Algorithms. But, we do not overload a server with high-bandwidth requests in our approach. Accordingly, we do not monitor an increase in dropped client requests. Therefore, as shown in Fig. 5, we reduce the request drops roughly 57% according

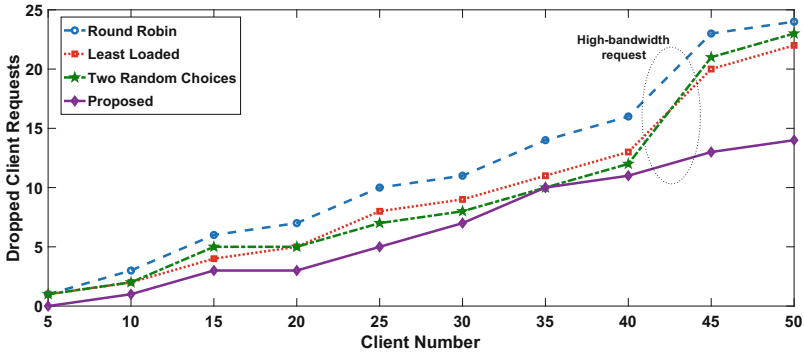


Fig. 6. Request drops due to queue overload

to other three algorithms. Also, in all evaluation results, the Round-Robin has the worst performance compared to the Least-Loaded and Two Random Choices algorithms because of the static characteristic.

5 Conclusion

In this paper, we proposed a Parametric-Decomposition based request routing in CDNs. With this method, we can combine the high-bandwidth client requests on origin server and split again to more than one surrogate server. We modeled the origin server with superposition and queuing procedures based on G/G/1 queuing system to estimate the load status. The load information of origin server is used for split decision of client requests. Moreover, the surrogate servers are modeled with splitting and adjustment procedures based on G/G/1 queuing system. The queue loads and waiting times of surrogate servers are used to calculate the splitter index. We adopt the splitter index parameter to determine the optimal content split sizes to the surrogate servers. According to the simulation results, the proposed approach reduces the load on surrogate servers by 42% compared to the conventional approaches. Also, the latency and request drops are decreased by 44% and 57% compared to the conventional approaches, respectively.

References

1. Pathan, A., Buyya, R.: A taxonomy and survey of content delivery networks. Technical report, GRIDS-TR-2007-4, February 2007
2. Bilen, T., Canberk, B., Chowdhury, K.R.: Handover management in software-defined ultra-dense 5G networks. IEEE Netw. **31**(4), 49–55 (2017)
3. Flach, T., et al.: Reducing web latency: the virtue of gentle aggression. In: Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM 2013) (2013)
4. Jia, Q., Xie, R., Huang, T., Liu, J., Liu, Y.: The collaboration for content delivery and network infrastructures: a survey. IEEE Access **5**, 18 088–18 106 (2017)
5. Manfredi, S., Oliviero, F., Romano, S.P.: A distributed control law for load balancing in content delivery networks. IEEE/ACM Trans. Netw. **21**(1), 55–68 (2013)
6. Dahlin, M.: Interpreting stale load information. IEEE Trans. Parallel Distrib. Syst. **11**(10), 1033–1047 (2000)
7. Carter, R.L., Crovella, M.E.: Server selection using dynamic path characterization in wide-area networks. In: INFOCOM 1997, Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, Proceedings IEEE, vol. 3, April 1997, pp. 1014–1021 (1997)
8. Mitzenmacher, M.: The power of two choices in randomized load balancing. IEEE Trans. Parallel Distrib. Syst. **12**(10), 1094–1104 (2001)
9. Khare, V., Zhang, B.: CDN request routing to reduce network access cost. In: 37th Annual IEEE Conference on Local Computer Networks, October 2012
10. Kee, H.S., Lau, P.Y., Park, S.K.: Peered-CDN through request routing peering system for video-on-demand. In: 2013 13th International Symposium on Communications and Information Technologies (ISCIT), September 2013, pp. 66–71 (2013)

11. Arumaithurai, M., Seedorf, J., Paragliela, G., Pilarski, M., Niccolini, S.: Evaluation of ALTO-enhanced request routing for CDN interconnection. In: 2013 IEEE International Conference on Communications (ICC), June 2013, pp. 3519–3524 (2013)
12. Fan, Q., Yin, H., Jiao, L., Lv, Y., Huang, H., Zhang, X.: Towards optimal request mapping and response routing for content delivery networks. IEEE Trans. Serv. Comput. **PP**(99), 1 (2018)
13. Gross, D., Shortle, J.F., Thompson, J.M., Harris, C.M.: Fundamentals of Queueing Theory, 4th edn. Wiley-Interscience, New York (2008)
14. Cece, F., Formicola, V., Oliviero, F., Romano, S.P.: An extended ns-2 for validation of load balancing algorithms in content delivery networks. In: 2010 Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, pp. 32:1–32:6 (2010)