# Content-Aware Load Balancing in CDN Network

Ziqi Cui, Yongxiang Zhao*, Chunxi Li, Yunpeng Song, Wenwen Li
School of Electronic and Information Engineering
Beijing Jiaotong University
Beijing, China
e-mail: {18120047, yxzhao, chxli1,17120113, 19120085}@bjtu.edu.cn

2020 IEEE 6th International Conference on Computer and Communications (ICCC) | 978-1-7281-8635-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICCC51575.2020.9345240

*Abstract*—**Content delivery network (CDN) is an important infrastructure to transfer video traffic which is a major component of internet traffic. One hot topic in CDN field is load balancing mechanism, i.e. redirect users to CDN servers to efficiently utilize CDN resource. Most of related works try to provide available bandwidth or reduce delay to user as much as possible according to the CDN servers' heterogeneous available bandwidth or distance to users. However, recent works find that users' quality of experience (QoE) related not only to bandwidth and delay, but also to content of videos. Specifically, different video frames require different sizes to get the same quality of experience (QoE), and the gradients of frame size-QoE curves are various. Based on this observation, we propose a novel load balancing method in CDN by taking the above factors into account. Specifically, we build an optimization model and propose a corresponding heuristic method to explore the benefit of considering the influence of video content on QoE. The numerical results show that our method can improve users' overall QoE by more than 5% compared with the traditional methods.**

***Keywords- CDN, content-aware, load balancing, multi-sever, QoE***

## I. INTRODUCTION

Video traffic is a major component of the Internet traffic. According to Cisco report[1], video will account for 82% of Internet traffic by 2022, which means a million-minute video is delivered over the Internet per second.

Content Delivery Network (CDN) is an important approach to deliver video traffic. By pushing popular videos to edge caches closer to users, CDN reduces the transmission delay, and eases the burden of ever-growing data traffic on the backbone. Many video content providers such as Tencent, iQiyi, and YouTube leverage CDN to deliver high-quality videos to users.

Load balancing [2], i.e. matching users and servers, is a pivotal technology in CDN. On the one hand, CDN deploys a group of edge caches, geographically distributed over the networks to reduce the transmission delay to users, which means a user can be served by multiple edge caches and an edge cache may serve many users. On the other hand, the bandwidth used to carry video traffic of a server is limited since CDN operators need to rent bandwidth from Internet Service Providers (ISP). Consequently, there is a contradiction between the uneven distribution of users on the geographic plane and the limited bandwidth. For example, a large group of users may request video services from the same server so that the bandwidth of this edge cache cannot meet the needs of these users. Thus, matching users and edge caches in CDN is a key technology to fully utilize all edge cache to maximize the quality of experience (QoE).

Most of the existing works about load balancing in CDN try to increase available bandwidth or reduce delay to users as much as possible by delicate mapping users to edge cache servers. For example, the scheduler in [3] chooses a server and a peer group for the user based on the topology distance and geographical distance in order to reduce the transmission delay. In [4], the authors choose a server for the user based on server's load and the topological distance.

However, the QoE not only relates to network transmission states, such as bandwidth and delay, but also relates to the video content itself [5] [6]. Since the spatial and temporal complexity of the scenes are various, different video frames require different size (i.e. bitrate) to get the same quality QoE, and the gradient of the frame size-QoE curves is also different. This inspires us the load balancing policy should consider the above feature. For example, we can mix the frames with different frame size sensitivity into one edge cache to improve the bandwidth utilization and the overall QoE.

Based on this observation, we propose a content-aware load balancing algorithm in this paper. Specifically, we formulate the load balancing problem as an optimization problem aiming at maximizing the overall QoE, and propose a heuristic method to solve it. Finally, we evaluate the performance of our algorithm with numerical experiments.

The remaining paper is organized as follows. Section II presents the related works about CDN load balancing. In Section III, we briefly introduce the scenario of our system and our motivation. In Section IV, we formulate the load balancing problem as an optimization problem, and the heuristic algorithm is proposed in Section V. The performance evaluation is shown in Section VI. Finally, we conclude this work.

## II. RELATED WORK

Load balancing in CDN is critical to fully utilize network resources and improve QoE. There are many works on this topic, and they can be classified as the following types.

One type tries to balance the load among CDN servers to prevent overload of specific servers and provide stability service to users. In [7], the live platform, twitch, redirects user's request to the server with the smallest requests in one region and dynamically balances users among regions to prevent overload. The authors in [8] model a Lyapunov optimization where the quadratic Lyapunov function and the drift reflects network congestion and the traffic loads respectively, and solve the optimization problem to avoid traffic overload.

The second type takes video QoE into consideration. In [3], the scheduler chooses a server and a peer group for the user based on the topology distance and the geographical distance. The authors in [9] show that users with different engagement in live broadcast platforms provide various profits to the platform and prove that higher platform profit can be gained by allocating more bandwidth to users with higher engagement. In [2] [10] [11] [12], the proposed methods dynamically select a server for users by monitoring network link status such as Round-Trip Time (RTT), bandwidth, etc.

The third type jointly considers traffic load and QoE to map users to proper servers. In [4], the authors jointly consider each server's load and the topological distance between the user and servers to select the optimal server. In [13], the authors combine service experience, load balancing, and traffic overhead to optimize server selection. The study in [14] shows that YouTube takes the RTT to the data center as the most crucial parameter, meanwhile, load balancing and content popularity are also considered.

All the works above try to delicate server selection mechanisms and improve QoE by balancing the server load, reducing transmission delay, or increasing transmission bandwidth. However, video content itself is also a critical factor, which is ignored by all of these researches. In this paper, the influence of video content on QoE is considered by jointly deciding server selection and video bitrate to maximize users' QoE.

## III. Scenario and Motivation

Application scenario is shown in Fig.1. In this figure, there are cache servers with limited upload bandwidth and unlimited storage resource to provide video service to users, and numerous users requesting videos from these cache servers.

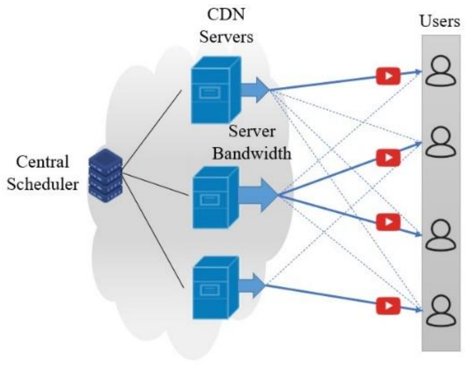
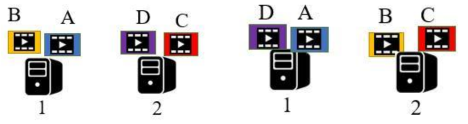


Figure 1. Application scenario

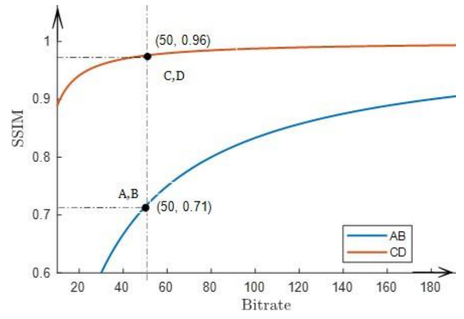
We first introduce some research results about QoE. Structural Similarity (SSIM) [15] is a common metric of video QoE which use luminance, contrast and structure to reflect the perceptual quality of video frames. Recent research [6] shows that the relationship between SSIM and frame size can be described as a convex function and the parameters are different for different frames, which means that the SSIM of different frames has different sensitivity to frame size.

Based on this observation, we propose a novel load balancing method in CDN by considering the difference in QoE sensitivity of frames. Fig.2 gives an illustrative example of the basic idea of our method. As is shown in the figure, 4 frames, A, B, C and D need to be uploaded to users by servers 1 and 2 at the same time. The QoE-bitrate curves of A, B, C and D are shown in Fig.2(c), in which curves of A and B coincide, so is C and D. We also assume that both upload bandwidth of servers 1 and 2 is fixed to 100 unit.

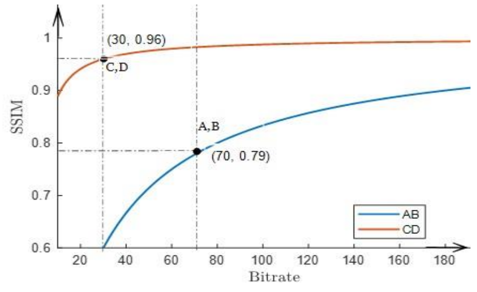
There are two schemes to assign these four video frames to two servers. Assigning a frame to a server means this frame will be uploaded by the server. In the first case, frames A and B, C and D are assigned to servers 1 and 2 respectively, as shown in Fig.2(c). In this case, each frame is allocated 50 unit bandwidth because of symmetry property, and the overall SSIM is 3.34. In the second case, A and D, C and B are assigned to server 1 and 2 respectively. The total SSIM can reach 3.5 if we assign 30 unit to C and D, 70 unit to A and B. This example shows that the total SSIM can be improved by selecting the combinations of frames and frame size wisely when the bandwidth of servers is limited.



(a) Assignment 1 (b) Assignment 2



(c) Frame SSIM of assignment 1



(d) Frame SSIM of assignment 2

Figure 2. Two assignment methods and the corresponding SSIM

## IV. System Model

In this section, we build an optimization model to maximize users' overall QoE by selecting the combinations of frames and frame size wisely.

The observation period is discretized into $T$ slots, and the slot set is indicated as $\boldsymbol{T}$, where $\boldsymbol{T} = \{1,2,\ldots,T\}$. A slot duration is fixed to a frame duration, and it can be easily extended to a larger time interval such as chunk duration to ease the computational burden [5]. We consider a network with $M$ cache servers, and denote the server set as $\boldsymbol{M} = \{1,2,\cdots,M\}$, and each server has enough storage capacity to cache all the video frames. $\boldsymbol{B} = \{B_1, B_2, \cdots, B_M\}$ denotes the bandwidth of cache servers, where $B_i$ represents the $i^{th}$ server can upload $B_i$ unit bitrate per slot. $\boldsymbol{N} = \{1,2,\ldots,N\}$ represents the frame set. The number of requests of $i^{th}$ frame in $t^{th}$ slot is denoted as $W_i^t$. $f_i(r)$ represents the relationship between the bitrate $r$ and the SSIM of the $i^{th}$ frame, and this function can be calculated offline. According to [6], $f_i(r)$ can be fitted by $1 - 1/(ar + b)$, for $a > 0, r > -b/a$, where parameter $a$ and $b$ may be different for each frame.

At the end of each time slot, the central scheduler receives the frame ID requested by users in the next slot, then maps frame requests to proper servers, and determines the size of frames by scheduling the optimal model proposed in this section. A binary variable $I_{ij}^t$ $(t \le T)$ represents the mapping decision in $t^{th}$ slot, in which $I_{ij}^t = 1$ represents the requested frame $i$ is uploaded by server $j$, and $I_{ij}^t = 0$, otherwise. And $r_i^t$ represents the size of $i^{th}$ frame at $t^{th}$ slot.

The symbols and their definitions are summarized as Table 1.

TABLE I. Symbol Definition

| Symbol | Definition |
|---|---|
| $\boldsymbol{M} = \{1,2,\cdots,M\}$ | The server set. |
| $\boldsymbol{N} = \{1,2,\ldots,N\}$ | The frame set. |
| $\boldsymbol{B} = \{B_1, B_2, \cdots, B_M\}$ | The servers' bandwidth set. |
| $\boldsymbol{IDLE} = \{idle_1, \cdots, idle_M\}$ | $idle_i$ is the idle bandwidth of $i^{th}$ server which is the difference between the server capacity $B_i$ and allocated bandwidth. This value is initialized as $B_i$. |
| $\boldsymbol{Y} = \{(i,j,r), \ldots\}$ | The decision set, each element is an ordered tuple, tuple $\{i,j,r\}$ represents the $i^{th}$ frame is assigned to $j^{th}$ server with size $r$. |
| $\boldsymbol{W^t} = \{W_1^t \cdots W_N^t\}$ | The request set, $W_i^t$ is the request number of $i^{th}$ frame in $t^{th}$ slot. |
| $\boldsymbol{U_n}$ | The unassigned frame set. |
| $r_i^t$ | The bitrate of $i^{th}$ frame in $t^{th}$ slot. |

A frame is uploaded by one of the servers, and delivered to the users who have requested it by P2P or multicast. Thus, we impose that,

$$\sum_{j=1}^{M} I_{ij}^t = 1, \quad \forall i \in \boldsymbol{N}, \forall t \in T \tag{1}$$

And the bandwidth constraint of server $j$ at time slot $t$ can be expressed as (3), which means the overall bitrate of frames served by server $j$ in time slot $t$ cannot exceed the bandwidth limitation $B_j$.

$$\sum_{i=1}^{N} I_{ij}^t r_i^t \le B_j, \quad \forall j \in \boldsymbol{M}, \forall t \in \boldsymbol{T}. \# \tag{2}$$

Combining (1) and (2), the following optimization model is built to maximize the overall QoE in time slot $t$.

P1:

$$\max_{\boldsymbol{R},\boldsymbol{I}} \sum_{j=1}^{M} \sum_{i=1}^{N} W_i^t I_{ij}^t f_i(r_i^t)$$

s.t.

$$\sum_{i=1}^{N} I_{ij}^t r_i^t \le B_j \# \tag{C1}$$

$$\sum_{j=1}^{M} I_{ij}^t = 1 \# \tag{C2}$$

$$I_{ij}^t \in \{0,1\} \# \tag{C3}$$

$$r_l \le r_i^t \le r_u \# \tag{C4}$$

P1 is an optimization problem to maximize the overall QoE provided by all edge cache servers in $t^{th}$ slot under the cache servers' link capacity constrain, where $W_i^t$ is the request number of $i^{th}$ frame, and $W_i^t I_{ij}^t f_i(r_i^t)$ is the obtained overall QoE of the $i^{th}$ frame at $j^{th}$ cache server when its frame size is $r_i^t$. (C4) limits the bitrate lies in the range of $r_l$ and $r_u$, which is the minimum and the maximum bitrate respectively.

P1 is a mixed-integer problem hard to solve with a large number of frames and servers. In the next section, a heuristic algorithm is proposed to solve the optimization problem.

## V. Heuristic Algorithm

The upper bound of our algorithm performance can be found by aggregating all servers' bandwidth on a virtual server. In this case, all the frames are assigned to the virtual server, and the above integer mixed model will degenerate to a convex model that is easy to solve. We name it as virtual model.

Under the guideline of the virtual model, we propose the following heuristic algorithm. Firstly, the virtual model is used to find all unassigned frames' optimal size. Secondly, a greedy algorithm is conducted to allocate the unassigned frames with the size calculated according to virtual model to actual servers. After that, the steps are repeated till all the frames are assigned to proper servers. Finally, if there is still idle bandwidth in a server, it will be equally allocated to the frames in that server.

### A. *The Ideal Virtual Model*

In the ideal virtual model, we calculate all the unassigned frames' size with the virtual server whose bandwidth is the sum of all servers' idle bandwidth. The optimization problem is modeled as P2.
P2:

$$\max_{\boldsymbol{R},\boldsymbol{T}} \sum_{i\in \boldsymbol{U_n}} W_i^t\, f_i(r_i^t)$$

$s.t.$

$$\sum_{i\in \boldsymbol{U_n}} r_i^t \le \sum_{j}^{M} idle_j \# \qquad (C1)$$

$$r_l \le r_i^t \le r_u \# \qquad (C2)$$

where, $idle_j$ is the idle bandwidth of $j^{th}$ server which is the difference between $j^{th}$ server capacity $B_j$ and allocated bandwidth. $\boldsymbol{U_n}$ is the unassigned frame set.

### B. *Greedy Allocation*

After obtaining each unassigned frame's size using the above virtual model, we try to allocate each unassigned frame with calculated size to a proper actual server using the following greedy policy.

For each unassigned frame $i$, we define its value $v_i$ as the ratio of the overall QoE it can provide and its size as (3).

$$v_i = W_i^t f_i(r_i^t)/r_i^t \# \qquad (3)$$

Then the following test is conducted to allocate unassigned frames to actual servers. Firstly, we sort the unassigned frames and servers according to the frame's value and the idle bandwidth respectively in descending order, and name the sorted results as frame list and server list. Then we check whether the bandwidth of the target server (Top-ranked server in the server list) is larger than the size of the target frame (Top-ranked frame in the frame list). If the answer is yes, then the frame is allocated to the server with its calculated frame size in the virtual model, then we update the unassigned frame set and the idle bandwidth of the server. If the answer is no, we update the upper bound of frame size as the maximum idle bandwidth, then repeat the above ideal virtual model to recalculate the optimal size of unassigned frames, then execute the greedy allocation again. We will repeat the above ideal virtual model and greedy allocation if the unassigned set is not empty and there is any idle bandwidth.

Algorithm 1 shows the details of the above steps. In this algorithm, line 4 uses the virtual model to calculate the size of unassigned frames and lines 5-16 use the greedy algorithm to allocate frames to servers. Line 17 allocates the remaining bandwidth equally to the frames allocated to the same server.

Our algorithm can be divided into two main steps. First, we calculate the size of frames using the idea virtual model by some optimization algorithm, like interior point, active set, etc. We use $O(opt)$ to denote the complexity of the first step. Then we calculate the value of each frame, and find out the target frame and the target server, their sort complexity is $O(NlogN)$ and $O(MlogM)$ respectively. Assuming in the worst cases, we repeat the two steps every time we assign a frame, so we repeat the two steps at most $N$ times. Therefore, the complexity of our algorithm is $O(N * max(O(opt), O(NlogN), O(MlogM)))$. In general, $O(opt) > O(N) > O(M)$, so the final complexity is $O(N * O(opt))$.

**Algorithm1 Allocation policy**

```
Input: server band $\boldsymbol{B}$, frame set $\boldsymbol{N}$, request set $\boldsymbol{W}$
Output: Decision list $\boldsymbol{Y} = \{(i,j,r_i)\ (k,p,r_k) \ldots.\}$
1:  Initialize: $\boldsymbol{Y}$=$\emptyset$, $\boldsymbol{U_n} = \boldsymbol{N}$, $\boldsymbol{IDLE} = \boldsymbol{B}$
2:  While $\boldsymbol{U_n} \neq \emptyset$
3:      Calculate $r_i$ for each $i \in \boldsymbol{U_n}$ with virtual model
4:      Calculate value of each frame $v_i = W_i f_i(r_i)/r_i$
5:      For $i \in \boldsymbol{U_n}$ do
6:          sort unassigned frames in descending order of
            value;
7:          For $j \in \boldsymbol{M}$ do
8:              sort servers in descending order of idle
                bandwidth;
9:              If $r_i$<=$idle_j$ then
10:                 Y ←$(i, j, r_i)$, $idle_j \leftarrow idle_j$- $r_i$,
11:                 Remove $i$ from $\boldsymbol{U_n}$
12:                 Go to 5
13:             End if
14:         End for
15:     End for
16: End while
17: Allocate the remaining idle bandwidth of each server
    equally to frames assigned to that server.
18: Return Y
```

## VI. Numerical Results

In this section, we use numerical experiments to evaluate the performance of our scheme.

We choose 10 raw movies from different categories in Tencent video app and encode each video into five copies with different video quality and frame sizes. Then we use the MSU Video Quality Measurement Tool [16] to calculate each frame copies SSIM values under different frame size. Finally, the function $f_i(\cdot)$ can be obtained by fitting five pairs of SSIM and corresponding size of each frame. In the following experiments, we assume that a time slot duration is a frame duration.

The raw video files we used are 720p, and the other copies are trans-encode by the raw video, so we define the frame size of 720p as unit one. The normalized size of other copies is the ratio of their size to that of 720p frame.

We randomly select 200 different frames from 10 movies as the frame set $\boldsymbol{N}$. We also use Zipf distribution to generate the value of $W_i^t$ , which is request number of $i^{th}$ frame, and set the overall request number to be 1000. Furthermore, we set 10 cache servers with the same bandwidth. The algorithm also can be used in the situation where the servers' bandwidth is not equal.

We compare our scheme with the following baselines.

Random policy: This policy randomly dispatches the video frames to servers, and the frames provided by the same server share the bandwidth equally of that server. This method corresponds to the scheme of Domain Name System (DNS), where the server randomly redirects a user to an edge cache, it is commonly implemented on the Internet.

Evenly policy: This policy assigns video frames evenly to servers. This scheme corresponds to the typical load balancing mechanism.

Popularity-aware policy: This policy considers frames' popularity. In the first step, each server is assigned equal user requests number. In the second step, we assign frames to servers to ensure that the sum of the requests number of frames allocated to a server is equal to the request number calculated in the first step. Specifically, we allocate the frame with the largest request number to the server with the most available request number.

Ideal policy: This policy treats all servers as one server whose bandwidth is the sum of all servers' bandwidth, and uses the ideal virtual model to calculate the size of frames. This policy obtains the upper bound of performance.

We sequentially number the above policies for convenience. We compare our scheme with a baseline by the percentage of performance improvement as follows.

$$(QoE_{heuristic} - QoE_{baseline})/QoE_{baseline} \# \qquad (4)$$

Where, $QoE_{heuristic}$ and $QoE_{baseline}$ is the QoE of heuristic algorithms proposed in this paper and one of baselines listed above, respectively.

## A. Impact of Server Bandwidth

First, we consider that the servers have the same bandwidth resource and change the bandwidth value to observer QoE. Due to each frame's request number being equal, the policy 2 and 3 are equal in this condition.

In this experiment, we fix the number of servers to be 10, and assume all servers' bandwidths are the same. We vary servers' bandwidth from 10 to 500 with step of 50. We also set the request number of each frame to be the same. For each value of bandwidth, we calculate the QoE of the heuristic algorithm and the 4 baselines, then use (4) to find their performance improvement ratio, the results are shown in Fig.3.

In Fig.3, the x-axis is the bandwidth value, and the y-axis is the relative QoE improvement ratio. As is shown in the figure, our scheme is always better than policies 1, 2 and 3. It can be observed that our scheme has more significant performance when the bandwidth is less than 200 while the performance improvement diminishes as server bandwidth increases further. It is because the video SSIM is more sensitive to the change of frame size when the bandwidth is insufficient, and SSIM will be saturated when the bandwidth is sufficient. However, the gap between our scheme and other baselines remain almost unchanged, it maintained at about 5% for policies 2 and 3, and 10% for policy 1.

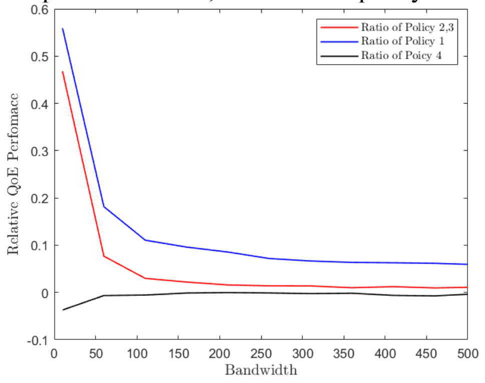


Figure 3. Numerical result when bandwidth is equal

## B. Impact of Frame Popularity

We consider the influence of frame popularity in this subsection. We fix the bandwidth of each server as 100, and the number of servers as 10. In each experiment, we generate request numbers of each frame following Zipf distribution under the condition that the total number of users is 1000. We repeat the experiment by varying the parameter $\alpha$ of Zipf distribution from 0.1 to 1 with step 0.1.

Fig.4 shows the numerical results when varying $\alpha$. It can be seen that our scheme has a little gap compared to the ideal policy. It can also be seen that our policy performs much better than policies 1, 2, and 3, because they neglect the influence of the request numbers of frames and QoE-frame size relationship. Policy 3 considers the request numbers in frame assignment. However, it does not consider the QoE-frame size relationship. As a result, its relative performance to our policy stays around 5% under different α.

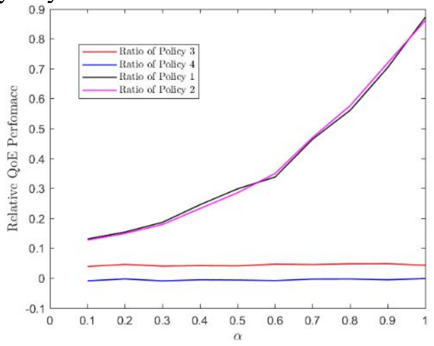


Figure 4. Numerical result when varying Zipf parameter

## C. Impact of Server Numbers

In this experiment, we fixed the total bandwidth of all servers while changing the number of servers from 2 to 30, and all servers equally split the total bandwidth. Moreover, we set the request number of each frame to be the same, so policies 2 and 3 are the same. We repeat the above experiment when the total bandwidth is 500 and 1000 respectively.

Fig.5 shows the relative performance improvement compared to policies 2, 3 and 4 under different server numbers. The impact server numbers on QoE performance are almost negligible when changing the server number. Fig.5 also shows that our scheme works better when the sum of bandwidth is insufficient (B=500), and the improvement is less obvious when the sum of bandwidth is 1000.

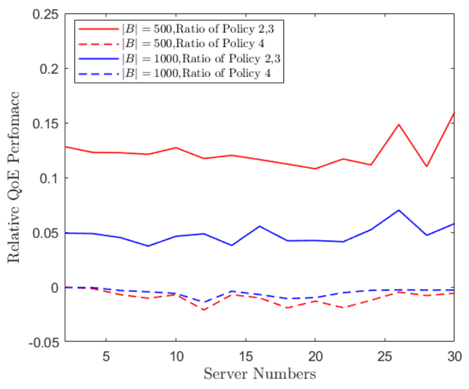


Figure 5. Numerical result when varying server number

## VII. Conclusion

CDN is a practical approach to deliver video traffic. However, its bandwidth has always been a bottleneck in service performance improvement. CDN load balancing is critical to fully utilize the scarce bandwidth resources and provide satisfying QoE to users. In this paper, we design a CDN load balancing algorithm considering the QoE-frame size relationship and schedule video request among CDN servers. We formulate this problem as an optimization problem and propose a heuristic method to solve it. The numerical results show that our method can improve the overall QoE by about 5% compared with the traditional method.

## Acknowledgment

This work was supported in part by National Natural Science Foundation of China under Grants 61872031.

## References

[1] "Cisco Visual Networking Index: Forecast and Trends, 2017-2022 White Paper," [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service/provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] V. K. Adhikari et al., "Measurement Study of Netflflix, Hulu, and a Tale of Three CDNs," IEEE/ACM Transactions on Networking, vol. 23, no.6, Dec. 2015, pp. 1984-1997.

[3] A. Saengarunwong and T. Sanguankotchakorn, "A Two-Step Server Selection in Hybrid CDN-P2P Mesh-based for Video-on-Demand Streaming," 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2018, pp. 499-504.

[4] T. Drazen, Z. Drago, "Dynamic Server Selection by Using a Client Side Composite DNS-Metric," Tehnicki Vjesnik - Technical Gazette, 2018, pp.1080-1087.

[5] Y.Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang and C.Yue, "ABR streaming of VBR-encoded videos: Characterization chanllenge and solutions" , Proc. 14th Int. Conf. Emerg. Netw. Exp. Technol., 2018, pp. 97-108.

[6] D. Ray, J. Kosaian, K.V. Rashmi, S. Seshan., "Vantage: Optimizing video upload for time-shifted viewing of social live streams," SIGCOMM 19: Proceedings of the ACM Special Interest Group on Data Communication August 2019, 2019, pp. 380-393.

[7] J. Deng, G. Tyson, F. Cuadrado and S. Uhlig, "Internet scale user generated live video streaming: The Twitch case," International Conference on Passive and Active Network Measurement, 2017, pp. 60-71.

[8] J. Liu, Q. Yang and G. Simon, "Congestion Avoidance and Load Balancing in Content Placement and Request Redirection for Mobile CDN," in IEEE/ACM Transactions on Networking, vol. 26, no. 2, April 2018, pp. 851-863.

[9] W. Yang, Y. Hu, L. Ding and Y. Tian, "Viewer-Oriented CDN Scheduling on Crowdsourced Live Video Stream," 2019 IEEE 2nd International Conference on Electronics and Communication Engineering (ICECE), Xi'an, China, 2019, pp. 112-117.

[10] C. Fangfei, S. Ramesh K and T. Marcelo, End-user mapping: Next generation request routing for content delivery, Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, pp. 167-181, August 17, 2015.

[11] H. Tran, S. Souihi, D. Tran and A. Mellouk, "MABRESE: A New Server Selection Method for Smart SDN-Based CDN Architecture," in IEEE Communications Letters, vol. 23, no. 6, June 2019, pp. 1012-1015.

[12] B.Shilpa and T.Venkatesh, "An overlay management strategy to improve QoS in CDN-P2P live streaming systems," Peer-to-Peer Networking and Applications, v 13, n 1, January 1, 2020, pp. 190-206.

[13] W. Ting, S. Junde Song and Menia, "A three-stage global optimization method for server selection in content delivery networks," Soft Computing, 2017, pp.467-475.

[14] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting video server selection strategies in the YouTube CDN," in Proc. 31st Int. Conf. Distrib. Comput. Syst., 2011, pp. 248-257.

[15] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, April 2004, pp. 600-612.

[16] MSU Video Quality Measurement Tool, [Online]. Available: http://www.compression.ru/video/quality measure/video measurement tool.html