Load Balancing Strategy for P2P VoD Systems

Guimin Huang[1], Chengsen Li[1], Pingshan Liu[1,2]
[1]Research Center on Data Science and Social Computing, Guilin University of Electronic Technology
Guilin, China
[Email: sen_5201@163.com]
[2]Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology,
Guilin, China
[Email: 32015581@qq.com]

Received December 10, 2015; revised April 24, 2016; accepted July 1, 2016;
published September 30, 2016

Abstract

In a P2P (Peer-to-Peer) VoD (video-on-Demand) streaming system, the nodes' load is an important factor which affects the system performance. In the system, some nodes may receive too many requests, which leads to overload. On the other hand, some other nodes may receive too few requests, which leads to low utilization. Therefore, designing a reasonable load balancing strategy is important. However, existing related studies cannot handle this problem effectively, because they don't have an efficient dynamic load information management mechanism, and they don't distinguish the difference of requests when transfer the nodes' load. In this paper, to manage the dynamic load information efficiently, we design a load management table for each node. Based on the load information, we propose a load balancing strategy which uses a request migration algorithm (LBRM). Through simulations, our scheme can handle the load imbalance problem effectively and improve the users' playback fluency.

Keywords: load balance, Peer-to-Peer, video-on-demand, request migration, load transfer

A preliminary version of this paper was presented at the International Conference on Computer Science and Technology (CST2016). This work was supported by the the Foundation of Key Laboratory of Cognitive Radio and Information Processing, Ministry of Education (GUET, No. CRKL150105), the Science and Technology Research Program of Guangxi University (No.KY2015ZD047), the Foundation of Key Laboratory of Guangxi Trusted Software (No. kx201409), the Foundation of Guangxi Experiment Center of Information Science (No. LD13076X), the Foundation of Guangxi Programs for Science and Technology Development (No. GuiKeGong 1598019-3).

1. Introduction

In recent years, the P2P VoD streaming systems have been deployed widely. However, due to the arbitrariness behavior of nodes, some nodes may receive too many requests. Thus, in the process of nodes handling these requests, some requests may be delayed or can not handled in time, while some other nodes may receive too few requests which lead to low utilization. In this case, the load imbalance problem occurs, which can leads to system performance degradation and impair the users' experience. At the same time, due to the nodes in the network may be heterogeneity, the handling capacity or bandwidth of each node is different, this leads to the request pressure on each node is not the same, and some nodes may invalid due to overloading. So how to keep the load balance between nodes in order to avoid that some nodes become overloading, how to ensure the requests from other nodes to be responded more stable and faster, these are the problems we need to solve.

The load balancing technology is a good solution to these problems [1]. Nevertheless, as far as we know, the factors considered in the present study are not comprehensive enough. For example, most of the research make use of tracker server to manage the node's load information, but excessive reliance on server will aggravate the server load and bring about network congestion. In addition, some studies have not distinguished the difference of the requests when transferring the node's requests, i.e., time-bounded and urgency of requests, as the high degree of emergency request may be transferred mistakenly, which affect the user's playback quality.

Therefore, we propose a new load balancing scheme which use a request migration algorithm to balance the load between the nodes. In our scheme, mainly includes the contents are the following: (1). We constructed a load information management table of nodes, which can effectively avoid the use of tracker server to manage and propagate node load information. (2). Analyzing the difference of the received requests. Because of we comprehensive consider the priority and deadline of the requests, we transfer requests to low utilization nodes successfully and ensure users' play fluency at the same time. (3). We construct a node's utility function to rational select the nodes with high performance and stability as the object of load transfer, it can help us ensure the reliability of the load transfer.

The paper is organized as follows. In Section 2, we briefly introduce some related work. In Section 3, we explain our load balancing strategy in detail. In Section 4, we evaluate the performance of our proposed algorithm. Finally, we conclude this paper in Section 5.

2. Related Work

At present, many scholars have done a lot of efforts to solve the problem of load balancing, and a number of load balancing strategies have been proposed to improve the performance of P2P systems. In order to manage node load information in whole network, K. Graffi et al. [2] proposed to use a specific node which called responsible peer to manage the node information of a part of nodes, but information exchange with responsible peer frequently might cause the single-point bottleneck problem. Similarly, literature [3] utilized super nodes to manage node information which can incur the same problem. In reference [4], the author presented a load balancing algorithm based on partial network information to improve system load balancing speed, but this algorithm is based on the assumption that the capacity and ability of each node

is consistent, which doesn't conform to the actual situation. Bharambe AR et al. [5] estimated global load information in the network using random walks, however, in a system containing N nodes, in order to estimate the load distribution more accurately, the algorithm requires the use of O ($logN$) random walks to support the calculation, which would increase the complexity of the algorithm and the network overhead.

In order to balance the load in the network effectively, a self-adaptive load balancing algorithm was proposed in [6], in which nodes would create binary tree back-up node tables for their shared hot files automatically, and transfer extra query request to back-up nodes. Nevertheless, it does not consider the difference of the requests to be transferred, which may cause that some requests miss the deadline and do harm to the stability of the system. Yanming Shen et al. [7] employ SQS policy which chooses the peer with least load to send request so as to avoid some peers receive too many requests. However, Alix L.H.Chow et al. has already revealed in their research [8] that this mean is difficult to implement, as it requires exact knowledge of instantaneous node queue lengths, although periodically updating the node's request queue size is also not accurate. Literature [9] proposed a multi-attribute range query approach that also incorporates load balancing, it creates approximate histograms using random sampling to manage nodes, but introduce a large computational overhead. In the literature [10], an incentive m echanism based on game theory was proposed to improve the situation of uneven load among nodes, but the algorithm needs to obtain global network information in general, it is hard to achieve.

3. LBRM Algorithm

In this section, we first design a load information management table, then we describe the algorithm we proposed. For a quick reference, we list the main symbols used in the paper in **Table 1.**

Table 1. List of symbols used in the paper

Symbol	Description
ID	node identifier
R_j	the scarcity of the request which required data piece j
T	a time period that the node record data
B	the total upload bandwidth of the node
N	the total neighbor nodes of the required node
U_j	the urgency of the request asking for data piece j
t_q	the time that corresponding the requested data piece's playback position
t_c	the node's current playback time
P_j	the priority of the request asking for data piece j
$f(t)$	the online probability starting from the node logging on to moment t
$F(t)$	cumulative online probability value
S_i	the maximum upload speed of node i
P_i	the total amount of upload data pieces of node i
W_i	utility value of node i

3.1 Node Information Management

In order to manage and disseminate the load information of each node easily, we construct a node information management table. This table does not save the load information of all the nodes in the network, but only save the node and its neighbor nodes' information. we define

every twenty nodes within the neighbors as a node cluster, and there are a number of clusters in the whole network. Then a node information table will be generated based on each node cluster, it is used as the definition of node load and the basis of load transfer. This table is divided into two parts: the upper part is mainly used for the judge of load state, the data only depends on the local information of the node itself. The data with the lower half part need to be derived from the network and used to record the load information of the neighbor nodes. ID is a node identifier. Then the parameters of our algorithm mainly come from this table, as show in **Table 2**.

Table 2. Load information management table

Peer ID	Type	Name	Size	Number	Time
	1	Data upload	32KB	n_1	07/29/01:02
	2	Confirm message	1KB	n_2	07/30/08:32
	3	Heartbeat message	12B	n_3	…
	4	Request message	1KB	n_4	…
	5	Connect message	1KB	n_5	07/30/09:21

Backup peer	Utility value	Load state
ID_1	n_1	L
ID_2	n_2	N
…	…	…
ID_{20}	n_{20}	O

3.2 Load Definition

In a P2P VoD system, the key issue to realize the dynamic load balancing of the nodes is identifying those overload nodes and transferring their load to other low utilization nodes. So, we must define the load of nodes firstly. Because the bandwidth resource is one of the most important index in P2P system, it is closely related to nodes' own bandwidth to upload data or download video files. In this paper, we use *loadDegree* to define the load of nodes. The greater value the *loadDegree* is, the greater pressure the node bear. It is calculated as follows:

$$loadDegree = \frac{\left(\sum_{t}^{t+T} us + \sum_{t}^{t+T} ms\right) \times 8}{B \times T} \tag{1}$$

Among them, us is the total amount of video data uploaded by the node, which belongs to the type 1 in **Table 2**. ms is the size of all kinds of message packets sent by the node, which is in response to the type 2 to 5 in **Table 2**. T is a period of time and B is the total upload bandwidth of the node. Thus, the calculation of the node load only depends on the local information and needn't to get other information from the network so as to calculate more conveniently and save the cost.

3.3 Load Decision

Obviously, how to determine whether the node is overloaded and how the load information is transmitted in the network when nodes are aware of their own load by local information is the key problems of this stage. We solve it by the following two sections:

3.3.1 load state classify

To design a dynamically changing node load information table is a big challenge because that the load of a node is a time varying value. We divides the load into three grades as shown in **Table 2**. It is denoted as state *O* when *loadDegree* is greater than 85%. Similarly, it is denoted as state *L* when *loadDegree* is less than 30%, the rest is denoted as state *N*. It is unnecessary to update the load change in real-time after dividing the node state. We just need to pay attention to the state switch of the node(for example: from *N* to *O*), and the load information is updated once there is a state switch. Generally, in the steady state of P2P network, 66% nodes have no contribution to the whole system and 20% nodes provide 98% sharing files [11], this means that only a small number of nodes switch their state in a time unit, we don't need to update too much information, so we can greatly reduce network overhead in this way.

3.3.2 load information spread

In P2P VoD system, we call those nodes which have the same resource as neighbors, they exchange BM(buffer map) with each other periodically. When a data request was sent to a node in a node cluster, the requested resource is also contained in other nodes in this cluster, so we only transfer the load of the nodes that within the node clusters. That is to say, we need only spread the load information between the nodes in the cluster. We use biased gossip protocol to form a network view based on these node clusters, and the message is disseminated and updated only in corresponding nodes cluster when a node's load state changes.

By this means, the resource node which need to transfer the load is aware of the load information of the neighbor nodes. we can manage the node's load information without using the tracker server or a specific management node, and it can effectively reduces the overhead of load information disseminate in the network by classifying the load information.

3.4 load Transfer

The intended purpose of this section is to transfer the load to the low utilization nodes without affecting the user's playback fluency as much as possible. In order to reduce node load, we transfer a certain amount of queuing requests to those low utilization nodes in time, this can not only effectively alleviate the burden of the current node, but also guarantee the data requests to get a more quick response. However, if the overload node transfers those requests without screening, although it can decrease the load of this node, but some urgent requests may be transferred by mistake so that the request can not be timely responded. Therefore, we should consider not only the load of the node but also the priority of the transfer request in the process of transferring load so as to ensure that the emergency request isn't lost. Our algorithm is to measure the priority of the request by using the scarcity and urgency of the data piece. Firstly transferring these high priority requests that the node can not handle in time, then gradually transfer the low priority request.

First, we analyze the definition of the request's scarcity, as shown in formula (2):

$$R_j = 1 - \frac{\sum_{k=1}^{N} BM[k][j]}{N} \tag{2}$$

Here, scarcity refers to the proportion of the number of nodes caching a piece of data, which accounts for the proportion of the total number of the neighbor nodes. R_j indicates the scarcity of the required data piece j, N represents the total neighbor nodes of the required node. BM[k][j] indicates whether the neighbor node k missed data piece j or not, if k lost data piece j when it was caching the video resource which contained j, then we denote BM[k][j]=0, or BM[k][j]=1. Thus, the more nodes that lost this data piece in the network, the sooner we should handle it, so as to reduce the rarity of the data and improve the efficiency of resource sharing.

Then we deduce the following formula in respect of the urgency of the request:

$$U_j = 1 - \frac{t_q - t_c}{T} \tag{3}$$

Here, U_j indicates the urgency of the request asking for data piece j , t_q represents the time that corresponding the requested data piece's playback position, t_c represents the node's current playback time, T represents the total duration of the video. We can know from formula (3) that the more t_q close to t_c, which means that the more urgent the data request is. According to formula (2) and formula (3), we can get the request priority as shown in formula(4):

$$P_j = U_j * R_j = \left[1 - \frac{\sum_{k=1}^{N} BM[k][j]}{N} \right] * \left[1 - \frac{t_q - t_c}{T} \right] \tag{4}$$

P_j indicates the priority of the request asking for data piece j .After get the priority, we can gradually transfer the requests with low priority out contrasting to the value of P_j until the node are not overloaded. But in the P2P VoD system, not all the high priority requests can be successfully processed. So we combine the request deadline to screen out some urgent requests which cannot handle in time. Here, we introduce an example as follows:

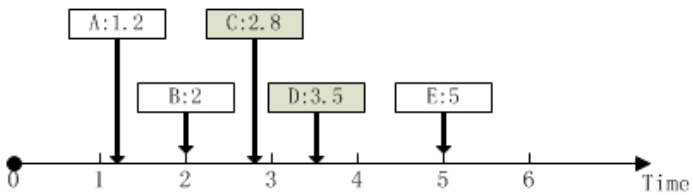

Fig. 1. Initial ordering of requests

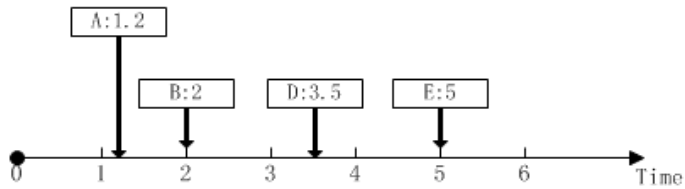

Fig. 2. Ordering after processing

We assume that the time for each request handled by the node is one time unit, and normalized the deadline of each request. We ordering the requests within five time units by the deadline (as shown in **Fig. 1**). For example, the time required for the successful handling of the request *A* is one time unit, due to the deadline of *A* is 1.2 unit, which is more than one unit, so request *A* can be successfully processed by the node. Likewise, request *B* can be successfully processed, too. We need two time units to deal with *A* and *B*. That is to say, when we finished processing *A*, *B* and *C* we need 3 units time. However, because of the deadline of *C* is 2.8 unit, which is less than 3, it means that the node can't handle *C* in time, similarly, *D* can't be treated in time due to the lack of time after handling *C*. So we have to make a prejudgment before a period of time, to remove *C* from the request queue and transfer it to other low utilization nodes. So we can do it like this:

Step 1. Ordering the queuing requests within 5 time units by the deadline.

Step 2. Determine if there is more than one request in a time unit, if not, turn to step 4.

Step 3. Determine whether the time required for the successful handling of the request is small than the deadline of the request, if the answer is yes, then we judge the next request, or we need to transfer the request.

Step 4. Continuing to filter the next time unit, if this five time unit are already screened, then continue to the next round of the iteration, turn to step 1.

We can see from **Fig. 2** that *D* becomes a request that can be processed successfully after eliminating request *C* which can not be handled. By this means, firstly we transfer those high priority requests which can't handle in time, then we gradually transfer the requests with low priority out until the node are not overloaded. That is to say, we use the way which combines the priority with deadline to carry out the process of the request transfer.

3.5 Node Selection

Here, we discuss how to select the appropriate low utilization node as the loads' receiver. The focus is that we should to choose those nodes which have sufficient and stable bandwidth as the receiver of the load as much as possible. We use the utility function of the nodes as the basis for the selection here. However, Literature [12] pointed out that a peer does not automatically know its own available uplink bandwidth. In other words, If we still need to measurement the available bandwidth of the other nodes when we transfer the load, this is bound to affect the timeliness of information, lead to inaccurate measurement results, and introduce additional information overhead. In order not to have additional information exchange, we predict the node's ability to provide bandwidth by recording the historical maximum upload speed of nodes. Besides, regarding the stability of the node, we use the online time of nodes to herald it. Literature [14] found that in a P2P VOD system, the probability density distribution function of the online time of nodes obeys lognormal distribution. According to mathematical derivation, we can get:

$$f(t) = \frac{1}{\sqrt{2\pi}\sigma t}\exp\left[-\frac{(\ln t - \mu)}{2\sigma^2}\right], t > 0 \tag{5}$$

Here, $f(t)$ represents the online probability starting from the node logging on to moment t, μ represents the average value of the logarithm of the node's online time t, σ represents the standard deviation of the logarithm of the node's online time t. In order to illustrate the stability

of the nodes more clearly, we can calculate the cumulative online probability values of the node starting from it logging on the network to moment t by using the formula (5), namely:

$$F(t) = \int_{0+}^{t} f(t)dt \tag{6}$$

For node i, we also consider other factors, here we use S_i to represent the node's maximum upload speed. P_i represents the total amount of the node's upload data pieces. Then, the node's utility functions are as follows:

$$W_i = F(t) * (\alpha S_i + \beta P_i) \tag{7}$$

The bigger the value of $F(t)$, the longer the node's online time in the future. α and β is the weighting factor. $F(t)$ and P_i indicate the stability of the nodes, when the node is in a low utilization state. The greater the value of W_i, the more the service ability it can provide.

3.6 LBRM Algorithm

In this section, we briefly summarize our algorithm. The basic idea of this selection algorithm is that: according to the state switching rule described above, the node updates the load information within a node cluster when the node has just become an overloaded node. Beginning, the overload node give priority to select the nodes which with the state L as the transfer object. Among those nodes, the overload node will select a node with high utility value to transfer load. When the state of the chosen node shifted from L to N due to receiving load continually, we initiate a new round of state updating. The node updates the load information and the utility value of its neighbors, and we can select a new low utilization node to continue the process of load transfer. If there is no low utilization node in the neighbors, then we select the node in N state for transmission. When we initiate another round of state updating after the state is transferred from N to O, we should abandon this node and select another node which not overload for load transfer.

Algorithm 1 LBRM algorithm

```
collect the information of nodes
calculate the loadDegree
while loadDegree >85% do
 calculate P_j
 judge whether each request can handle in time
  if not then
    regard these requests which cannot handle as
          the object of earliest transfer
  else
    choose the requests with low priority as the object of transfer
  end if
  for each backup node do
    if the node's state is L then
     priority to choose this node as the load receiver
    end if
```

```
    if the backup node with state L isn't exist then
      choose the nodes with state N as receivers
        if the backup node with state N isn't exist then
          send a reject message to the requestor and
              let it choose a new resource node
        end if
      end if
    end for
    for each object of transfer do
      use utility function to select the appropriate
    receiver as a transfer target
      start to transfer the load
    end for
  end while
```

4. Performance Evaluation

In order to evaluate the performance of our algorithm LBRM, we do extensive simulations and make comparisons with two other schemes: SQS [7] and a self-adaptive load balancing algorithm(SALB) [6]. As everyone knows, playback fluency, average uplink bandwidth utilization, node load distribution and overload number proportion are a few more important performance indicators in load balancing. So in this paper, we mainly compare and analyze the performance of those four indicators.

4.1 Simulation Settings

In order to evaluate the performance of LBRM algorithm, we do a simulation experiments. The final experimental result is an average value of data which is obtained from running the simulation program five times with the same parameters. The simulation model we constructed containing a tracker server, a streaming server and many peers. The tracker server have saved the information of all nodes and other servers, meanwhile it is also responsible for returning the original list of the resource nodes, and the streaming server have cached all the video resources in the network, the request node will ask the streaming server for resources when other nodes in the network don't contain.

Our simulation parameters are set as follows. (1) The initial node number is 100 and the joining of nodes obeys the linear distribution, this way is similar to the rule of node joining in the literature [15]. In each period the number of the nodes is increased by 30 and the maximum size is 3000 peers. There are 500 shared videos in the network, the node's upload bandwidth is randomly set at a range of 70~200Kbps. (2) The upload bandwidth of the tracker server is set to 20Mbps, and the delay is 10ms. The upload bandwidth of the streaming server is set to 50Mbps. (3) The size of the node cluster is initialized as 20, and the node exchanges its BM(buffer map) messages with its neighbors in every 5s. (4) A previous study [16] has described that a peer needs to download some consecutive video data before it can start to play back. In order to pre-fetch data, we have to divide the video file. All videos are divided into blocks with the same size 2MB and the last block may be less than or equal to 2MB, then each block is divided into 64 pieces, and the size of each piece is 32K.

4.2 Simulation Results

In the P2P system, the nodes should contribute their own upload bandwidth resources. Since we transfer the load of the overloaded node to the idle low utilization node, so the bandwidth utilization of nodes in the network is relatively improved. We plot the CDF of the upload bandwidth utilization ratio in **Fig. 3** under three different schemes.

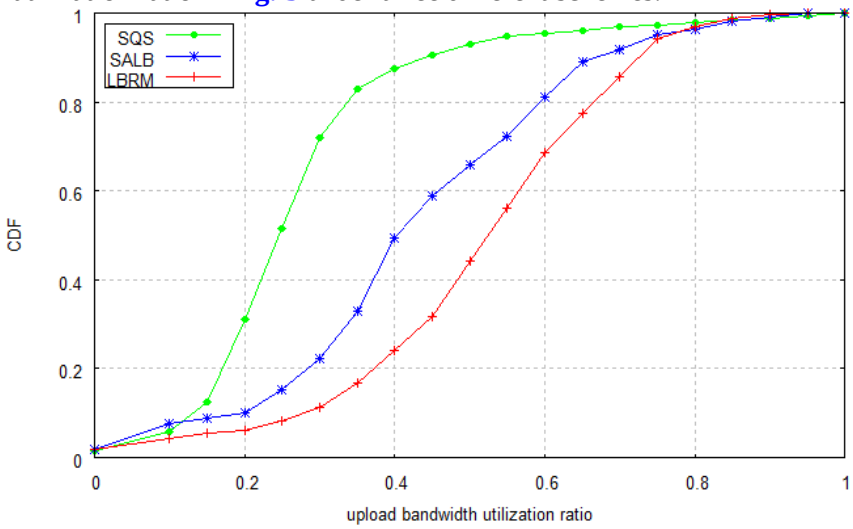

Fig. 3. the CDF of the upload bandwidth utilization ratio

From **Fig. 3** we can see, in the system using LBRM algorithm, only about 30% of the node's upload bandwidth utilization is less than 0.5, but the proportion has reached 95% and 63% when using SQS and SALB algorithm. The reason can be explained as that SQS algorithm send requests to the low load node preferentially, so most of the nodes' load in the network is in a relatively low level. Nevertheless, the optional objects for receiving loads are relatively less in SALB algorithm, so the number of nodes with low bandwidth utilization is more than that of our algorithm. Moreover, the total number of nodes using LBRM algorithm in the range of 0.6 to 0.8 of the bandwidth utilization is more than other two algorithms. From the graph, we can see that our algorithm is obviously superior to SQS and SALB.

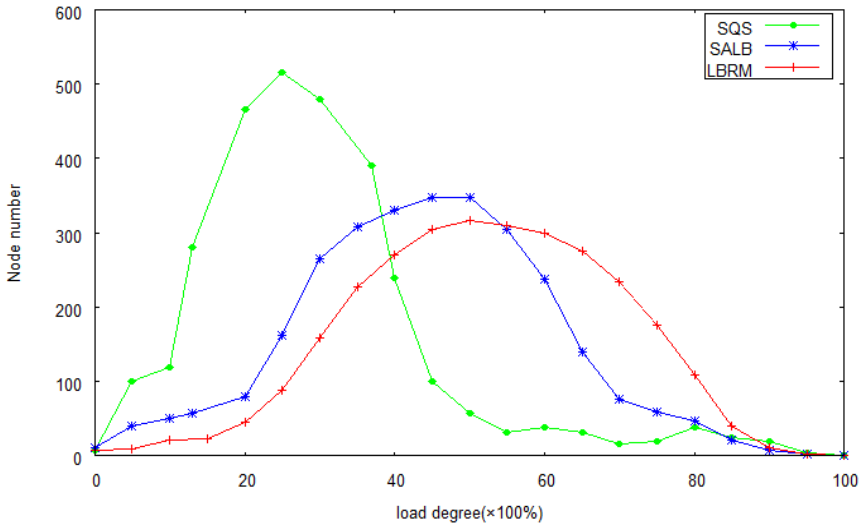

Fig. 4. the load distribution of the nodes

In order to better illustrate the load balance in the network, we list all the load distribution of the nodes. From **Fig. 4** we can see that the number of overloaded nodes in the P2P network using the SQS strategy is higher than the other two algorithms. The reason is that it doesn't set the threshold of node overload, and no further processing measures for overload nodes, which results in the generation of more overloaded nodes. And in the P2P network using the SALB policy, because of the overload threshold it set is the average value of the neighbor nodes, so the number of nodes in the low load area is larger, but the available bandwidth of those nodes with a strong ability is to be wasted in this way. In summary, the other two algorithms have poor performance in load balancing. But in the P2P network using the LBRM policy, thanks to our division of node states, most of the nodes' load is concentrated between 30% and 85%, which indicates that the load on each node is roughly balanced.

The playback continuity index(CI) under three different schemes is compared in Fig. 5. Playback continuity index is the ratio of the data actual obtained by the node to the data should be obtained in each time interval after the node begins to playback the video [13]. It reflects the fluency user playing video. The higher the value is, the more fluent the video playback. In the traditional P2P network, the overloaded node is unable to serve others in time, which results in that some request nodes can't obtain the data piece within a specified time, which greatly affects the quality of service(Qos). Therefore, a good load balancing strategy should be able to improve the user's playback fluency.

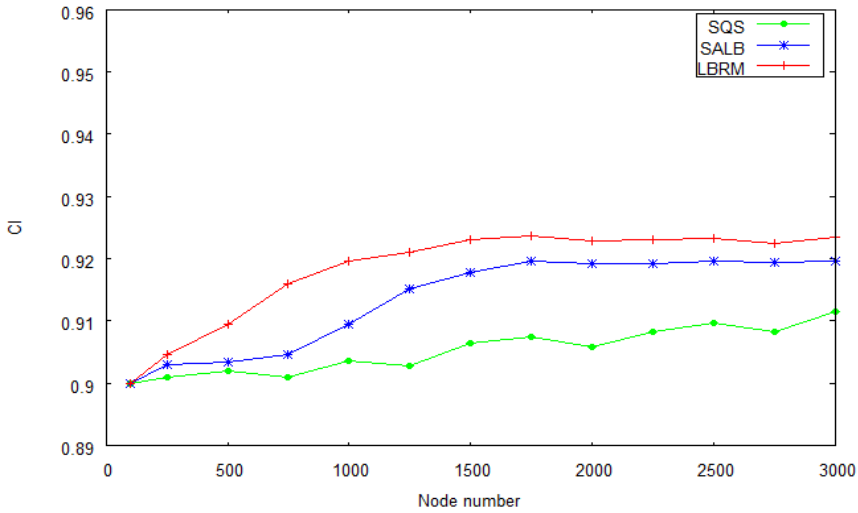

Fig. 5. the playback continuity index

From **Fig. 5** we can know, the three algorithms' quality of playback has improved with the increase of the number of the nodes which can also increase the number of the low utilization nodes, and it makes the transfer of load more convenient. When the number of nodes exceeds 1500, LBRM and SALB are basically reached a stable state, but the quality of SQS hasn't been improved too much yet, because it doesn't consider the emergency of the request and the receiver's bandwidth. Which makes the probability that the request can get a timely response isn't too high. Thus, we can get a higher degree of playback fluency when using LBRM.

Then we compared the performance of three algorithms on load balance degree. We measure it by the number of overloaded nodes here. The more the number of which algorithm's overloaded nodes in the network is reduced within a time unit, the more effective this load balancing algorithm is.

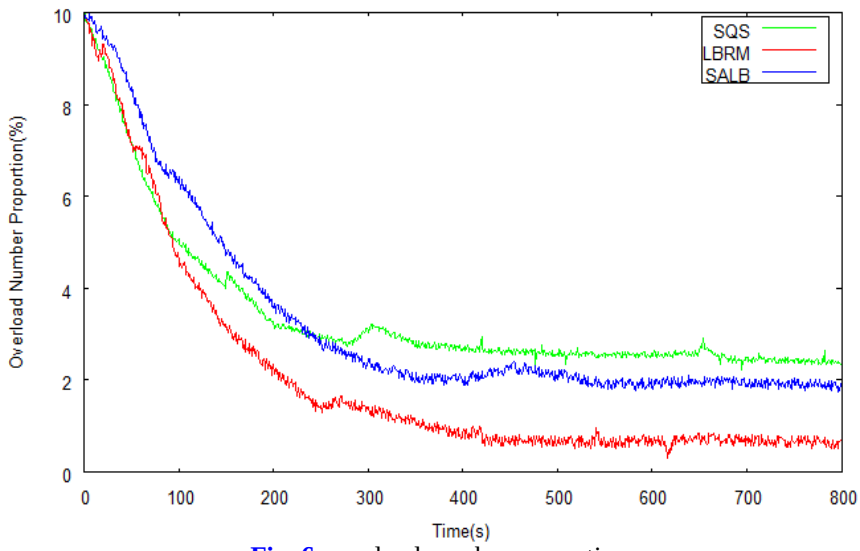

Fig. 6. overload number proportion

In **Fig. 6**, we start to record the data of the three algorithms when the number of overloaded nodes accounts for 10% of the total number of nodes in the network, within 0 to 200 seconds, the number of overloaded nodes was significantly declined in the network using LBRM and SQS, and the speed the number of overloaded nodes declining of SALB is slower than the first two algorithms. This is because LBRM is transferring requests directly, and SQS is sending a request to the low utilization node in a way of changing the request directly, so their improvement rate is larger. However, the SALB algorithm firstly needs to build a backup node of the popular shared files according to the request should be transferred, then the process of load transfer started, of course it has a poor efficiency. Within 300th to 500th seconds, due to SQS not considering the receiver's bandwidth and its' bearing ability, so the number of overloaded nodes is still more than our algorithm's, it is clearly that our algorithm is better than the other two algorithms in speed and efficiency in the aspect of reducing the overloaded nodes.

In order to test the performance of LBRM algorithm with different experimental parameters, we have adjusted the parameters of the experiment. Based on the original experimental parameters, we change the bandwidth distribution of nodes, and make the chum of system considered. The simulator determines each sender's uploading bandwidth following the distribution given in **Table 3**. This upload bandwidth distribution is proposed in a recent paper [17] according to various measurement studies on both corporate and residential users. Because the peers may use other Internet applications such as Web browser and e-mail when using the P2P streaming, the peers would not contribute all their upload bandwidth to the P2P streaming. The peers' lifetime follows an exponential distribution with a mean of θ s [18]. The value θ denotes the peer churn rate. The smaller the mean value θ, the higher the peer churn rate. In this experiment, we set the value θ to be 400s.

Table 3. Peer upload bandwidth distribution

Distribution(%)	10.0	14.3	8.6	12.5	2.2	1.4	6.6	28.1	16.3
Total upload bandwidth(Kbps)	256	320	384	448	512	640	768	1024	>1500
Contributed upload bandwidth	150	250	300	350	400	500	600	800	1000

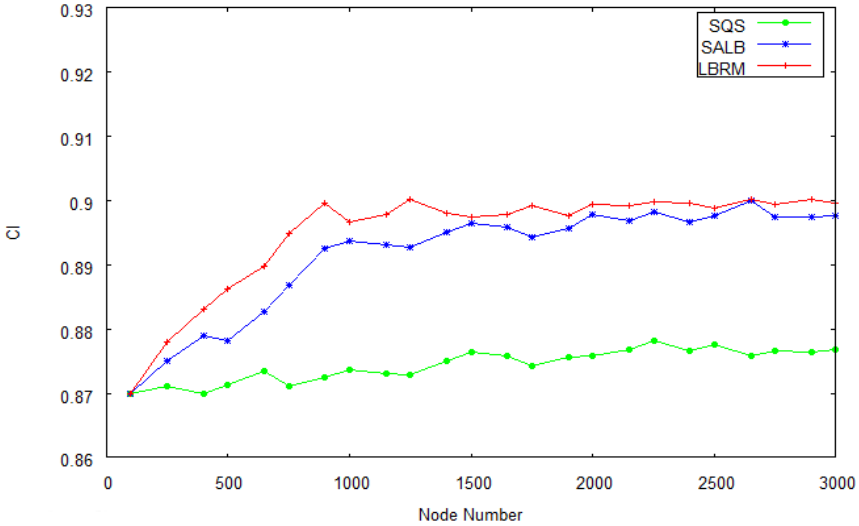

Fig. 7. the playback continuity index under different parameters

In **Fig. 7**, we compare the performance of the three algorithms in the playback continuity index once again. From **Fig. 7** we can see that the experimental results of these three algorithms have corresponding changes when the experimental parameters are changed. Because the churn is added in the system, the playback continuity index of the node is reduced, and the initial value is reduced to 0.87. In addition, the curve in the graph reflects the fluctuation of the network churn, which makes the playback quality of the nodes vary in a great range of fluctuations. In the initial stage, the growth rate of CI value of LBRM and SALB is higher than that of the SQS algorithm, the main reason is that the upload bandwidth of the node increases, which makes the number of nodes required for load transfer decrease. Therefore, the number of peers affected by the overload nodes reduces in system. However, SQS focuses on the number of nodes that receive the minimum number of requests, the effect of changing the nodes' bandwidth on the algorithm is very small. From the curve of each algorithm we can see, the LBRM algorithm has a better performance than the other two algorithms after changing the experimental parameters.

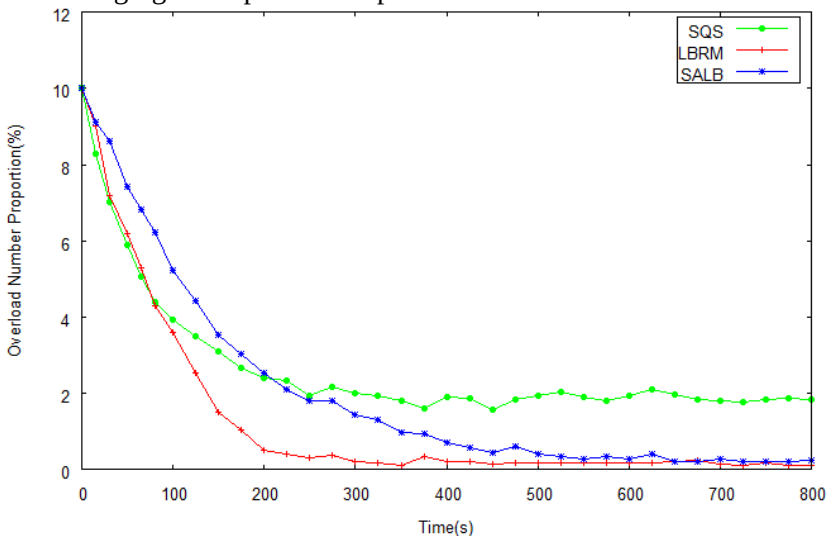

Fig. 8. overload number proportion under different parameters

In order to further observe the changes of the overload nodes in the network, we continue to record the ratio of the overloaded nodes. As shown in **Fig. 8**, From the beginning that we record the data, the suppression efficiency of the overloaded nodes with each algorithm is relatively high. However, after 200 seconds in the experiment, LBRM algorithm constantly requests redirection, which makes the ratio of the overloaded nodes almost become 0%. It can be seen that in the case of larger bandwidth resources, the operation effect of each load balancing strategy is more obvious.

5. Conclusion

In this paper, we have proposed a new load balancing scheme which uses request migration algorithm. This scheme mainly use in the P2P VoD system. First, we constructed a load information management table of the nodes. Through the information in the load information management table, we can manage the node's load effectively, which does not rely on the tracker server. Second, we proposed a new algorithm to reasonably transfer the requests from overload nodes to low utilization nodes. In our algorithm, we considered the emergency and the timeliness of requests during load transferring so that we can avoid the loss of data caused by the transfer of load in the traditional load balancing strategy. Third, we constructed a node's utility function and then selected the right node as the recipient of the request according to the utility value of the node. Through these measures, we have effectively solved the problem of uneven load in the network. Finally, we demonstrated the performance of this scheme through simulations. The simulation results showed that our scheme can effectively balance the load of nodes in the network.

References

[1] Qiao Y, Bochmann G, "Load balancing in peer-to-peer systems using a diffusive approach," *Computing*, 94(8-10): 649-678, 2012. Article (CrossRef Link).

[2] Graffi K, Kaune S, Pussep K, et al. "Load balancing for multimedia streaming in heterogeneous peer-to-peer systems," in *Proc. of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM, 99-104, 2008. Article (CrossRef Link).

[3] Zhong L, Xu C. "DLCA: Distributed load balancing and VCR-aware two-tier P2P VoD system," in *Proc. of Consumer Communications and Networking Conference*, 199-204, 2014. Article (CrossRef Link).

[4] Yao L, Dai G Z, Zhang H X, et al., "Load balancing algorithm for P2P systems based on partial network information," *Journal of Computer Applications*, 27(5): 1080-1082, 2007. Article (CrossRef Link).

[5] Bharambe A R, Agrawal M, Seshan S. "Mercury: supporting scalable multi-attribute range queries," *ACM SIGCOMM computer communication review*, 34(4): 353-366, 2004. Article (CrossRef Link).

[6] Xiong N, Xu K, Chen L, et al., "An Effective Self-adaptive Load Balancing Algorithm for Peer-to-Peer Networks," *Parallel and Distributed Processing Symposium Workshops & PhD Forum*, 1425-1432, 2012. Article (CrossRef Link).

[7] Yang S, Shen Y, Qu W, et al., "A Novel On-Demand Streaming Service Based on Improved BitTorrent," *Frontier of Computer Science and Technology, Fifth International Conference on. IEEE*, 46-50, 2010. Article (CrossRef Link).

[8] Yang Y, Chow A L H, Golubchik L, et al. "Improving QoS in bittorrent-like VoD systems," in *Proc. of INFOCOM, 2010 Proceedings IEEE*, 1-9, 2010. Article (CrossRef Link).

[9] Vu Q H, Ooi B C, Rinard M, et al., "Histogram-Based Global Load Balancing in Structured Peer-to-Peer Systems," *IEEE Transactions on Knowledge & Data Engineering*, 21(4):595-608, 2009. Article (CrossRef Link).
[10] Gupta R, Somani A K., "Game theory as a tool to strategize as well as predict nodes' behavior in peer-to-peer networks," in *Proc. of Parallel and Distributed Systems, 11th International Conference on. IEEE*, 1: 244-249, 2005. Article (CrossRef Link).
[11] Ying H, Zhigang C., "USMI: An Ultra-node Selection Mechanism with Incentive in P2P Network," *Multimedia Information Networking and Security*, 131-135, 2010. Article (CrossRef Link).
[12] Wang Y, Fu T Z J, Chiu D M. "Design and evaluation of load balancing algorithms in P2P streaming protocols," *Computer Networks*, 55(18): 4043-4054, 2011. Article (CrossRef Link).
[13] Vlavianos A, Iliofotou M, Faloutsos M., "BiToS: Enhancing BitTorrent for supporting streaming applications," in *Proc. of 25th IEEE International Conference on Computer Communications. Proceedings. IEEE*, 1-6, 2006. Article (CrossRef Link).
[14] Veloso E, Almeida V, Meira W, et al., "A hierarchical characterization of a live streaming media workload," in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet measurement*, 117-130, 2002. Article (CrossRef Link).
[15] Shen Y, Hsu C H, Hefeeda M., "Efficient Algorithms for Multi-Sender Data Transmission in Swarm-Based Peer-to-Peer Streaming Systems," *Multimedia IEEE Transactions on*, 13(4):762-775, 2011. Article (CrossRef Link).
[16] Liu P, Huang G, Feng S, et al., "Event-Driven High-Priority First Data Scheduling Scheme for P2P VoD Streaming," *Computer Journal*, 56(2):239-257, 2013. Article (CrossRef Link).
[17] Liu Z, Shen Y, Ross K W, et al., "Substream trading: towards an open P2P live streaming system," *Network Protocols*, 94-103, 2008. Article (CrossRef Link).
[18] Hei X, Liang C, Liang J, et al., "A Measurement Study of a Large-Scale P2P IPTV System," *Multimedia IEEE Transactions on*, 9(8):1672-1687, 2007. Article (CrossRef Link).

Guimin Huang is a full professor at Guilin University of Electronic Technology in China. He worked as an assistant professor at the Curtin University in Australia. Recently, he has published more than eighty academic papers on international journal and international conference, published one book of Peer-to-Peer network, awarded a patent of invention as well as five Software Copyright Registration Certificates. His research interests include distributed networks and text mining.

Chengsen Li is a postgraduate at Guilin University of Electronic Technology. He focuses on the P2P networks and its quality of service.

Pingshan Liu is currently an Associate Professor at Guilin University of Electronic Technology in China. His research interests include content distribution on streaming media, cloud computing and data mining.