Peer-to-Peer Netw. Appl. (2012) 5:312–322
DOI 10.1007/s12083-012-0140-z

EUE principle of resource scheduling for live streaming systems underlying CDN-P2P hybrid architecture

Chao Hu · Ming Chen · Changyou Xing · Bo Xu

Received: 29 November 2011 / Accepted: 30 April 2012 / Published online: 13 May 2012

Abstract Peer-to-peer live streaming offers plenty of live television programs for users, and has become one of the most popular Internet applications. However, some ubiquitous problems such as long startup delay and unsmooth playback seriously restrict the quality of service of live streaming, whereas deploying dedicated servers immoderately will suffer from excessive costs. In this paper, we introduce economical-underloaded-emergent (EUE) principle to instruct resource scheduling for live streaming systems based on CDN-P2P hybrid architecture. Complying with this principle, we differentiate peers' chunk requests according to their playback deadline and propose a set of mechanisms to provide distinct service for diverse requests. The results of simulation experiment demonstrate that EUE principle effectively optimize system performance, and achieve the remarkable reduction of startup delay and increase of chunk arrival ratio.

Keywords Live streaming · Design principle · Resource scheduling · CDN-P2P · Quality of service

1 Introduction

Recently, transmitting television (TV) programs over the Internet has become a more and more popular network application. According to incomplete statistic, this type of application has engendered a large-scale industry with tens of billions dollars worth of profits [1]. Despite the convenience of accessing Internet facilitates the transmission of television signal, it still brings insufficient bandwidth resources in partial network region if a great number of users watch TV programs at the same time. This situation leads to severe increase of packet delay and loss, especially when users and TV source are not located in the same Internet Service Provider (ISP).

To address these issues, content distribution network (CDN) was introduced to live streaming. CDN can achieve lower cross-region traffic and improve quality of service (QoS) of video streaming by deploying a great number of streaming servers. But this scheme has two main shortcomings, which are high cost and poor scalability. On the other hand, the emergence of peer-to-peer (P2P) presents us a scalable and economical approach for disseminating streaming video by exploiting the available resources in peers, but the intrinsic disadvantages of P2P such as weak capability and dynamic characteristic yet make P2P unable to provide QoS guarantee for users. In order to overcome the flaws of the two schemes, a consequential scheme combining CDN with P2P was proposed, and this scheme was expected to employ their respective advantages to construct a better streaming dissemination system. Although some research results show that this scheme has great prospects [2, 3], existing design approaches are yet unable to guarantee rigorous QoS requirements. Therefore, an inevitable problem for live streaming system is how to design a system based on CDN-P2P hybrid architecture and provide QoS guarantee for users.

In addition, Bo Li et al. measured the performance of a large-scale live streaming system called Coolstreaming [4] and found that nearly 5 % video chunks could not reach user's buffer before being played. Simultaneously, Yan Huang et al. analyzed the trace of PPLive [5] and declared that the buffering time of almost 20 % users occupied more than 80 % of the total time. In practical live streaming system, even a few chunk losses will make the screen frozen, and media players

C. Hu (✉) · M. Chen · C. Xing · B. Xu
Department of Computer Science and Engineering,
PLA Univ. of Sci. and Tech.,
Nanjing 210007, China
e-mail: huchaonj@126.com

have to wait until these chunks arrive at the user's buffer. This situation seriously worsens the quality of experience, thus it is very important to solve or alleviate the problem and devise a new design approach for live streaming system.

In this paper, we propose a novel system design scheme, namely Economical-Underloaded-Emergent (EUE) principle, to instruct resource scheduling for live streaming systems based on CDN-P2P hybrid architecture. Our contributions are as follows:

- From the perspectives of resource provision and consumption, we analyze the resource characteristics and user demands, and introduce EUE principle to guide the resource scheduling for live streaming to exploit system resources.
- Based on EUE principle, we differentiate peers' chunk requests according to their playback deadline and divide the urgent chunk requests into three classes, and then propose peer resource request scheduling mechanism to handle these chunk requests.
- We employ weighted fair queuing to allocate server resource, and determine the scheduling strategies and weight for each queue. Furthermore, we propose corresponding countermeasure to prevent the selfish nodes that try to get all the data from server.

The remainder of this paper is organized as follows. Section 2 outlines related work, and Section 3 proposes EUE principle for resource scheduling of live streaming. Section 4 introduces the peer resource request and server resource allocation mechanisms guided by EUE principle. Section 5 introduces experiment scenario and simulation results to evaluate the feasibility and efficiency of the proposed resource scheduling mechanisms. Finally, Section 6 presents a conclusion of our work.

2 Related work

In order to alleviate the pressure of streaming servers and exploit peer resources, P2P scheme was first introduced into a practical live streaming system in [6]. A data-driven based overlay network, namely DONet, was constructed to achieve efficient dissemination for live streaming. Moreover, AnySee [7] and PRIME [13] also adopted P2P for live streaming system design, and the exploit of peer resources effectively cut down the deployment costs of streaming servers. However, the intrinsic characteristics of P2P network, e.g. poor capacity and peer churn, as well as the impacts of firewall and network address translation (NAT), make these pure P2P based systems impossible to provide guaranteed QoS for users. In [2], CDN and P2P hybrid architecture was proposed to disseminate video streaming. In [3], a peer-assisted content delivery network was presented, and the architecture was divided into CDN distribution layer lying in the core network and P2P distribution layer locating in the access network, respectively. But these studies just focused on the framework design and lacked of in-depth resource scheduling analysis. In [14], a radically different cross-channel P2P streaming framework, namely View-Upload Decoupling, was proposed. This framework enabled cross-channel resource sharing to ensure resource provision is sufficient for user demands in each channel, but it introduced upload bandwidth overhead. In [20], the system performance of pull based P2P live streaming was analyzed, and the guideline of overlay topology design and chunk scheduling algorithm was provided to jointly improve user's QoS and reduce the costs.

In addition, improving scheduling strategies of live streaming is another way to raise user's QoS. In [10], a mixed strategy was proposed, and it achieved the tradeoff between startup latency and continuity. In [11], Yan Yang et al. introduced a deadline-aware scheduling, which included an earliest deadline first scheduling approach and an early drop based approach to address the service scheduling problem. In [16], random chunk and latest chunk were presented for chunk selection at the receiver. In [17], Silva et al. proposed to select receiver with a probability proportional to the peer upload bandwidth. In [8], a new streaming algorithm incorporating random network coding with a randomized push algorithm was employed to P2P live streaming. In [9], Nguyen et al. presented a novel design that combined network coding seamlessly with scalable video coding. Nevertheless, the above schemes neglect the characteristics of video chunks with different playback deadline, and can't provide differentiated service for diverse chunks.

There also have been many studies on performance analysis of live streaming system. In [22], Kumar et al. employed stochastic fluid theory to model P2P streaming systems, and exposed the fundamental characteristics and limitations. In [15], Bonald et al. studied the fundamental performance trade-offs of push-based diffusion schemes. In [18], chunk-based P2P video streaming was theoretically studied, and the delay bound to distribute video chunks was showed. In [21], Liu et al. analyzed the performance bounds of tree-based P2P live streaming, including minimum server load, maximum playback rate and minimum depth of the distribution trees. In [19], Abeni et al. presented the formal proof that there existed a distributed scheduling strategy being able to achieve the minimum steps to distribute a chunk.

3 EUE principle for resource scheduling

To describe our resource scheduling principle in detail, we firstly analyze P2P live streaming from the perspectives of resource provision and consumption.

Suppose *R(Server)* are the resources provided by CDN servers, *R(P2Poffered)* are the resources provided by all peers, and *R(LS)* is the overall resources. The system resources based on CDN-P2P hybrid architecture are composed of two parts, i.e., *R(Server)* and *R(P2Poffered)*. Thus

$$R(LS) = R(Server) + R(P2Poffered) \tag{1}$$

Every peer in the system must consume certain amount of resource to guarantee its QoS. Suppose the resource demanded for all users' smoothly viewing is *R(P2Pused)*, then system can't guarantee QoS for users no matter which scheduling scheme is used if Eq. 2 is satisfied.

$$R(LS) < R(P2Pused) \tag{2}$$

The system is likely to meet user's QoS demands by using a properly designed resource scheduling mechanism only if Eq. 3 holds.

$$R(LS) \geq R(P2Pused) \tag{3}$$

Therefore, only when Eq. 3 is satisfied, it is feasible to design appropriate resource scheduling mechanism to achieve user's QoS demand.

Due to dynamic and selfish behaviors of P2P users, *R (P2Pofferd)* and *R(P2Pused)* fluctuate in a certain bound, the following equation thus should be satisfied to meet Eq. 3 in a dynamic environment.

$$R(Server) + R(P2Pofferd)_{low} \geq R(P2Pused)_{high} \tag{4}$$

Where $R(P2Pofferd)_{low}$ represents the minimum amount of resource provided by P2P users, and $R(P2Pused)_{high}$ indicates the maximum amount of resource consumed by P2P users that can guarantee user's QoS.

For service providers, *R(Server)* is expected to be small so as to achieve more economic benefits. When $R(Server)=0$, the service providers do not need to spend any funds on server resources for live streaming system, but $R(P2Pofferd)_{low}>R(P2Pused)_{high}$ cannot always hold. So service providers must supply with enough server resources *R(Server)* to satisfy Eq. 4, and *R(Server)* is the most important factor to guarantee user's QoS. Therefore,

(1) P2P can decrease the system cost, but optimizing P2P resource scheduling mechanism is only able to alleviate QoS guarantee problem and can't eliminate it.
(2) CDN will increase the system cost, but CDN servers must be capable of providing sufficient resources when P2P resources can't be exploited any more.

Therefore, we propose three resource scheduling principles to design an economical but QoS-guaranteed live streaming system based on CDN-P2P hybrid architecture. These principles are as follows:

- *Economical principle*: P2P network resources should be exploited as many as possible.
- *Underloaded principle*: CDN servers should be endeavored to keep in light load state so as to deal with peers' emergent demand.
- *Emergent principle*: Emergent chunk that is close to playback deadline should be requested from the nearby CDN server. And servers should try their best effort to satisfy these emergent requests.

Economical principle indicates that live streaming system should endeavor to exploit the capability of P2P network to reduce the demand on CDN server resources. Underloaded principle indicates that CDN servers should stay in a light load state, and save their resources to provide the emergent requests with reliable service. Emergent principle indicates that peer should request resources from nearby server immediately once some emergent data appear, and then it will likely receive the video chunk in time. The three principles are called EUE principle.

4 Resource request and allocation mechanisms

4.1 Peer states and transition

In CDN-P2P hybrid architecture based live streaming, system resources come from servers and peers. The characteristics of server resources are stability and reliability, while peer resources can take on part of data delivery mission in despite of peer's low capacity, and resource allocation mechanism ought to assign system resources according to the feature of different resources. Therefore, to optimize user's QoS and system performance, server resources should be allocated to these urgent chunk requests to accelerate chunk dissemination or reduce chunk loss. Simultaneously, peer resources should be allocated to non-urgent requests to mitigate server's upload pressure.

From the perspectives of resource request, peers can request resources from servers or peers, and the requested object will alter in different situation. In order to explain this alteration, we divide the peer states into *original state*, *general state* and *urgent state*. Different state implies different peer's demands for system resources.

Original state refers to a condition that a peer just joins in live streaming system, and doesn't have enough chunks in its buffer for playback. At this moment, this peer desires to download video chunks to start playing live program. According to emergent principle, server should provide it with sufficient resources to reduce the waiting time. Meanwhile, based on economical principle, the resource provider should gradually evolve to other peers when peer receives enough chunks and starts playing.

General state refers to a condition that the absent chunks in peer's buffer are still relatively far from their playback deadline. According to economical principle and underloaded principle, live streaming should exploit the resources of P2P network to peers and keep servers in the light load state.

Urgent state refers to a condition that a peer doesn't receive the chunks whose playback time is close to the deadline, and these chunks will lose if they don't arrive at peer's buffer in time. Based on emergent principle, emergent resources should be supplied by servers.

Figure 1 shows a state transition diagram for peer states and their mutual transition in live streaming, where P2P will play the dominant role when peers are in general state.

4.2 Peer resource request mechanism

When a peer joins in the system, the peer should establish a neighbor relationship with streaming server to acquire some chunks from the server for playback. These sorts of chunks are called *I-type data*. After receiving enough chunks, peer should gradually request the subsequent video chunks from other peers; accordingly the load of I-type data on servers will be alleviated.

According to economical principle, a peer should seek resources from its neighbors if the playback deadline of chunk t_p satisfies $t_p > T_u$, where T_u is the maximum deadline of urgent chunks, and it is related with the size of urgent chunk buffer L_u. Since this requested chunk is not urgent, peer should select other peers as the chunk source. Obviously, the more powerful capacity a P2P network has, the less probability that general chunks become urgent chunks will be. But if $t_p \leq T_u$, the missing chunk will turn into urgent chunk, which is called *II-type data*, and peer must request resources from servers to download this chunk immediately. Otherwise, the chunk may be unable to arrive at the peer's buffer in time, so the parameter T_u is critical. On the one hand, if the size of T_u is set to be excessively large, the servers will have to bear greater burden and more server resources are needed. This results in the violation of

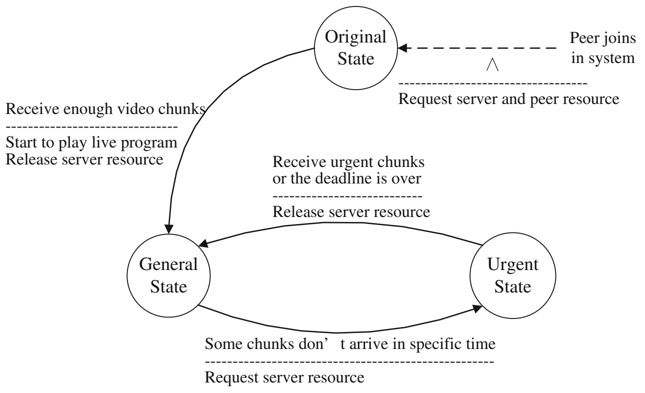
Original State
Peer joins in system
Request server and peer resource
Receive enough video chunks
Start to play live program
Release server resource
Receive urgent chunks or the deadline is over
Release server resource
General State
Urgent State
Some chunks don' t arrive in specific time
Request server resource

Fig. 1 Peer states and their transition in EUE principle

economical principle. If the size of T_u is much too small, the servers will take charge of lighter burden and more chunks will be provided by the peers, which complies with economical principle and underloaded principle but easily results in the violation of urgent principle. Figure 2 illustrates the peer resource request mechanism.

Besides I-type and II-type data, some fresh chunks generated recently by streaming server are scarcely possessed by peers, which are called *III-type data*. These III-type data could only be requested from server, whereas server resources will be wasted if all peers send chunk request for this type of data. So peers should decide whether they should request the chunk or not in a certain probability. Here we set the probability to be 0.1.

4.3 Server resource allocation mechanism

As mentioned earlier, servers should provide resources for the three types of urgent chunks. Providing resources for I-type data could enable peer receive sufficient chunks to start playing as quickly as possible, and promote user's QoS. The purpose of offering resources to II-type data is to ensure that chunks reach peer's buffer before they are played, which can keep the live program playing smoothly. Supplying peers with III-type data is to accelerate the dissemination of fresh chunks to prevent them from becoming II-type data. The functions of the three types of chunks are different, and their importance is also distinct, so servers should differentiate them when responding to these chunk requests.

In order to implement differentiated service, we employ weighted fair queuing (WFQ) to schedule server resources for the three types of urgent data. More specifically, for each type of urgent data, we build a queue and determine an appropriate scheduling strategy. In addition, each queue is assigned a weight factor, and the chunk requests are scheduled based on these weight factors. The work flow of server resource allocation mechanism is depicted in Fig. 3, where w_1, w_2 and w_3 are the weight of I-type data, II-type data and III-type data, respectively.

There are still two problems to be addressed. One is how to arrange the order of chunk requests in each queue, and the other is how to assign the weight for every queue.

4.3.1 Chunk scheduling strategy for each queue

For I-type data queue, peer will send chunk requests to server to require I-type data after joining in system, and the earlier request should be served firstly. So the first-coming-first-serve (FCFS) scheduling strategy could satisfy the requirements of I-type data queue.

In II-type data queue, all of these chunk requests are urgent, but the playback deadlines of these chunks have difference. Some chunks are going to be played immediately, while others still have some time. So the earliest deadline first scheduling

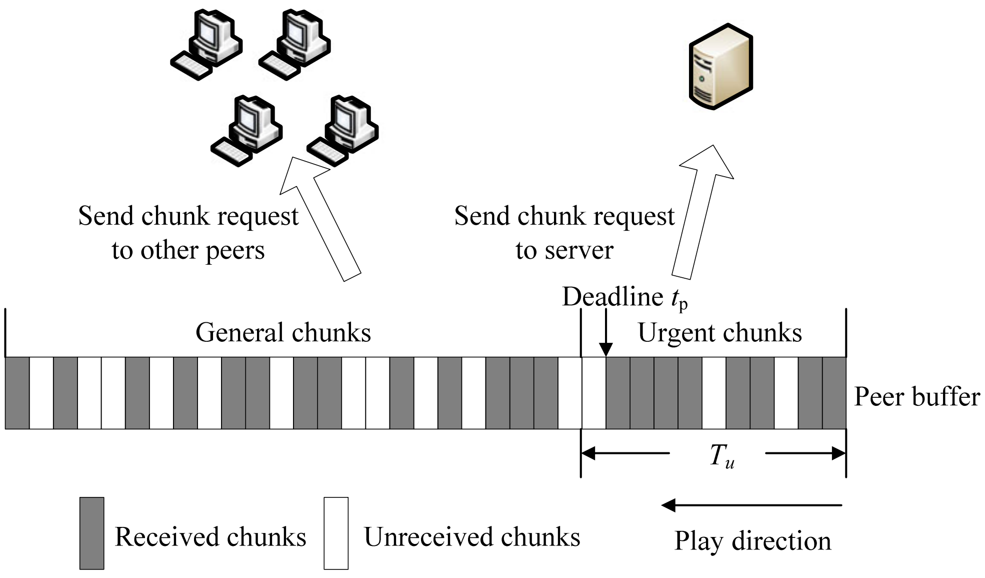

Fig. 2 Peer resource request mechanism

(EDF) strategy [11] is employed to arrange the sequence of chunk requests, and the request with the earliest deadline is served above all.

Suppose a new chunk request C enters II-type data queue at time T, and its playback deadline is D. At the same time, the chunk requests stored in II-type data queue are $[C_1, C_2, \ldots, C_n]$, and they are ordered according to EDF strategy. Furthermore, the playback deadline of chunk request C_j is D_j, and its arrival time is A_j. So the new request should be inserted after C_i, which must follow the Eq. 5.

$$D_i - (T - A_i) < D < D_{i+1} - (T - A_{i+1}) \quad (5)$$

If there doesn't exist any chunk C_i that satisfies the inequality D_i-$(T$-$A_i)<D$, the new chunk request should be inserted into the beginning of the queue. On the other hand, if the inequality D_i-$(T$-$A_i)<D$ is not satisfied under any circumstance, the request should be inserted into the end of the queue.

In III-type data queue, all the requested chunks belong to the fresh chunks, but the playback deadline is not urgent, so FCFS is adopted as the chunk scheduling strategy.

4.3.2 Weight assignment for each queue

When determining the weight for the three queues, the priority of the three types of urgent chunks must be considered, and the highest priority of queue should be assigned with the highest weight, so that these requests in this queue will have the best QoS, while other queues should assign proper weights to achieve some degree of QoS guarantee.

Specifically, if any chunk request in II-type data queue cannot receive service in time, the chunk may not arrive at peer's buffer before its playback deadline, which will lead to a significant QoS degradation. Because of the disastrous outcome, we assign the highest priority to II-type data queue. Simultaneously, the upper bound of queuing time in II-type data queue is set to be $T_u/3$, which enables peers have adequate opportunity to response to request timeout. The objective of reducing queuing time in III-type data queue is to improve the efficiency of chunk dissemination, which can cut down the amount of II-type data, so the priority of this queue is secondary. The maximum queuing time of chunk requests in III-type data queue is set to be T_u, which is equal to the maximum deadline of urgent chunks. When timeout happens, peers still have the chance to request these chunks from the server or other peers. Decreasing the latency of chunk requests in I-type data queue can shorten the startup time, and improve user's QoS, but the queuing time in I-type data queue is relatively elastic, so the priority of I-type data is the lowest. When server resources are deficient, the other two queues will be

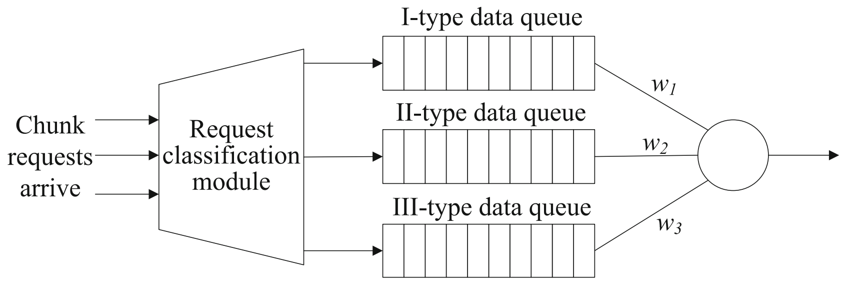

Fig. 3 WFQ mechanism for server resources scheduling

served preferentially. This resource allocation mechanism also implements access control. If the system resource is insufficient, this mechanism can prevent those peers that are watching live programs from being affected, e.g. the emergence of flash crowd. Furthermore, the upper bounds of the weight for II-type and III-type data urgent queues are prescribed to avoid the starvation of I-type data due to an excessively low weight. The maximum weight of II-type data and III-type data are defined to be M_2 and M_3, and the default values of M_2 and M_3 are 0.3 and 0.5, respectively.

Suppose the total server resources are R, the average size of a chunk is P, and the normalized weights of I-type data, II-type data, and III-type data queue are w_1, w_2 and w_3 respectively, where $\sum_{i=1}^{3} w_i = 1$.

In order to satisfy the waiting time bounds of II-type and III-type data, severs check the length of these two queues periodically and compute a new weight. The length of the queues are checked every time the server sends 100 packets. Assume $L_i^{'}$ and $L_i^{''}$ stand for the number of chunk requests in II-type and III-type data queue in the *i*th check respectively, while $\widehat{L}_i^{'}$ and $\widehat{L}_i^{''}$ are the estimated value in the *i*th check, respectively.

For II-type data queue, after the *i*th check, the resource allocated must satisfy $\frac{\widehat{L}_i^{'} P}{\omega_2 R} \leq T_u/3$, so,

$$w_2 = \min\left(\frac{3\widehat{L}_i^{'} P}{RT_u}, M_2\right) \tag{6}$$

Where $\widehat{L}_i^{'}$ is estimated by exponential weighted moving average, and the default value is μ=0.2. The formula for $\widehat{L}_i^{'}$ becomes

$$\widehat{L}_i^{'} = (1-\mu)\widehat{L}_{i-1}^{'} + \mu L_{i-1}^{'} \tag{7}$$

Similarly, we have

$$w_3 = \min\left(\frac{\widehat{L}_i^{''} P}{RT_u}, M_3\right) \tag{8}$$

Where $\widehat{L}_i^{''}$ is the estimated value after the *(i+ 1)*th checking III-type data queue, and $\widehat{L}_i^{''}$ is computed as Eq. 9

$$\widehat{L}_i^{''} = (1-\mu)\widehat{L}_i^{''} + \mu L_{i-1}^{''} \tag{9}$$

Finally, we obtain the weight of I-type data queue according to the following formula:

$$w_1 = 1 - w_2 - w_3 \tag{10}$$

4.4 Selfish behavior defense

In practical network environment, some selfish nodes may send a great deal of spurious urgent chunk requests and try to get all the data from server no matter they are in any peer state. These selfish behaviors will occupy most of server resources and seriously influence the operation of EUE principle, and the normal urgent chunk requests can't receive timely service.

In order to prevent these selfish behaviors, we propose a selfish node detection scheme to defend such malicious behaviors. In particular, servers periodically collect the statistical information, and record the amount of urgent chunk requests sent from each peer. Simultaneously, we set a threshold value N as the maximum number of chunk requests. Once a peer sends more than N requests to server in a period, it is deemed a selfish node, and then all of requests from this peer will be discarded.

5 Performance evaluation

5.1 Experiment scenario

To assess the system performance of resource scheduling mechanism proposed in the former section, we implement an event-driven live streaming simulator, which is based on the source code provided in [12]. The original simulator can simulate the packet-level data transmission and end-to-end delay among the peers. We modify this simulator, and enable the new one can support our peer requests and server resource allocation mechanisms based on EUE principle.

In our experiment scenario, there are one server and 2,000 peers in the system, and the streaming rate is 300 kbps. In addition, the upload bandwidth is the bottleneck, and all peers are classified based on their upload capacity. Table 1 shows the fractions of peers in different categories. Each peer has 15 neighbors and peers exchange their buffer maps periodically to perceive the chunks possessed by other peers and fetch the chunks by *pull* mode. To simulate the dynamic characteristic, some peers join and leave system randomly.

To evaluate the system performance and user's QoS under different resource scheduling mechanisms, the following metrics are used:

- *Startup Delay*: The metric refers to the interval from the time when a peer joins system until enough chunks have been downloaded and the live program starts to play. Obviously, the shorter the startup delay, the better the

Table 1 Upload capacity and fraction of peers in system

Upload capacity	Fraction of peers in system
768 kbps	0.15
384 kbps	0.3
128 kbps	0.55

user's QoS. In our simulation experiment, the peers start to play program after buffering video chunks for 15 seconds.

- *Chunk Arrival Ratio:* To guarantee continuous playback, the chunks must arrive at peer buffer before being played, otherwise, they are equivalent to missing, which will result in the degradation of playback quality. Chunk arrival ratio is defined as the value of these chunks arriving at peer's buffer in time divided by the total chunks that the peer should receive. The higher the chunk arrival ratio, the better the system performance.

5.2 Simulation results and analysis

5.2.1 The performance of resource scheduling mechanisms

To analyze the performance of resource scheduling mechanisms based on EUE principle, we compare these mechanisms (EUE for short) with traditional scheduling mechanism (TSM for short) that doesn't conform to economical principle, underloaded principle and emergent principle. In traditional scheduling mechanism, peers chooses neighbor randomly to construct an unstructured overlay, thus not every peer connects to server directly, but data exchange only happens among neighbors. In addition, peers in TSM select video chunk to download according to their playback deadlines, and the chunk with shorter deadline will be requested firstly. Figures 4 and 5 show the cumulative distribution function (CDF) of startup delay and chunk arrival ratio under different server bandwidth, respectively.

Figure 4 shows that startup delay in TSM has a great variance range, which changes from 15 seconds to 25 seconds. Furthermore, along with the increase of server bandwidth, the average startup delay decreases from 21.19 seconds to 20.9 seconds (about 1.37 %), which indicates scarce improvement. These results demonstrate that server resources are not fully exploited in this case, and startup delay cannot be reduced in despite of the double increase of server resources. In contrast, EUE can make use of the system resource more efficiently and reduce the average startup delay remarkably, which decreases from 21.36 seconds to 20.45 seconds (about 4.26 %). Moreover, the variance range of startup delay in EUE is also clearly less than that in TSM. We think this is because III-type data in EUE comes from server in startup phase, and server has a steady upload capacity. On the contrary, the chunks in TSM are mainly from peers, and the variant available upload bandwidth of peers has great impact on the stability of peer's downloading.

Figure 5 indicates that there are quite a number of chunks lost in TSM, and only about 60 % of peers have a chunk arrival ratio over 95 %, which implies that TSM cannot

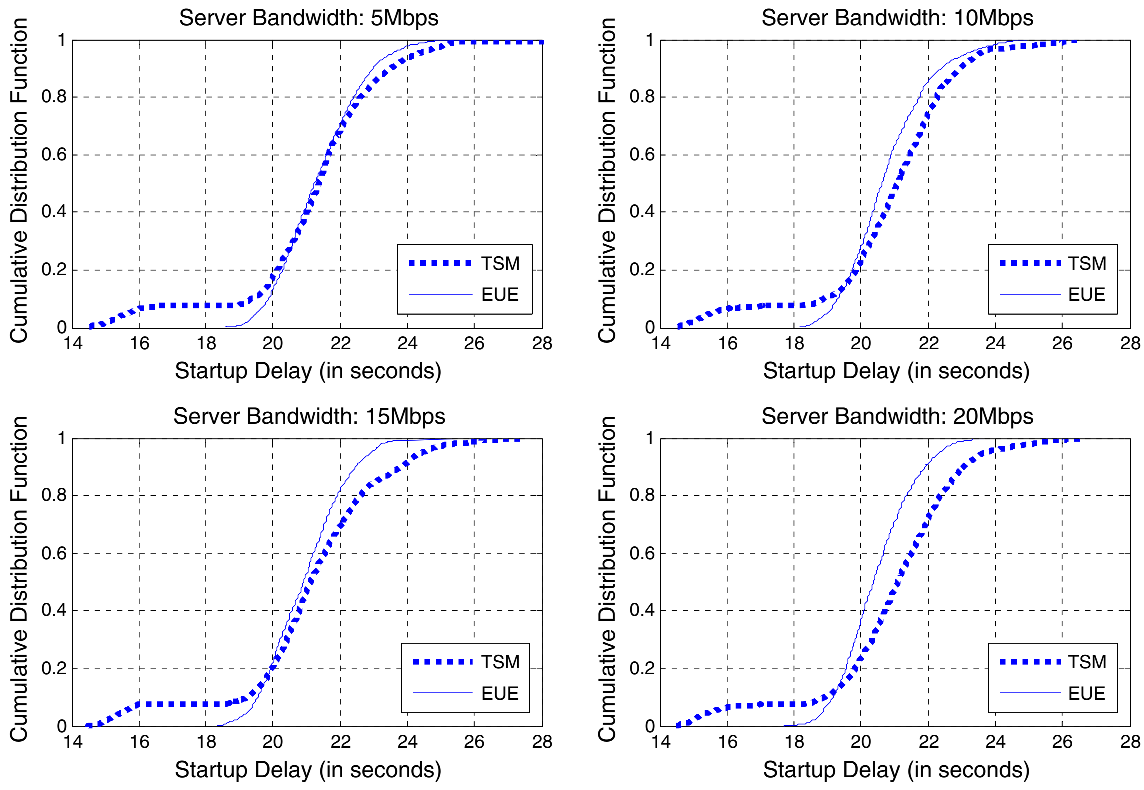

Fig. 4 The CDF of startup delay

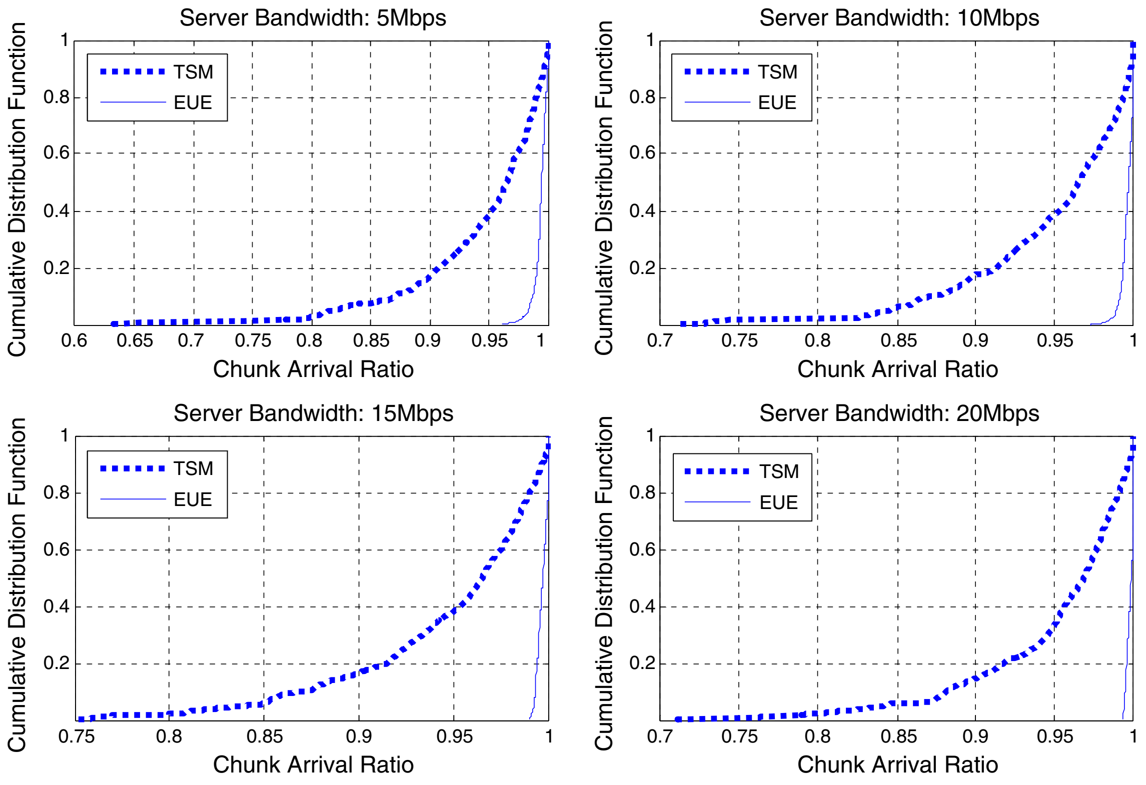

Fig. 5 The CDF of chunk arrival ratio

achieve high chunk arrival ratio. However, EUE can guarantee that the chunk arrival ratio of almost all peers is more than 98 %, and in particular, the average chunk miss ratio is less than 0.2 % when there are sufficient resources. The results demonstrate that servers can provide urgent chunks with guaranteed resource in EUE, which can reduce the amount of missing chunks and greatly improve user's QoS.

5.2.2 The impact of urgent buffer size

The size of urgent buffer T_u is critical for our EUE principle, because it directly affects the determination of requested object. If the value of T_u is set to be excessively large, the probability that ordinary chunk turns into II-type data will increase, and more chunks have to be requested from servers. However, server's capacity is limited, and too much load will incur serious chunk loss and worsen user's QoS. On the other hand, T_u implies the time limit for II-type data, and these urgent requests must be served before their playback deadline. During a peer requests II-type data from server, it has to wait for three parts of latency. The first part is the latency spending on request transmission from peer to server, and the second part is the queuing time that the request waits for service in server's queue, and the final part is the latency of video chunk sent from server to peer. So excessively small T_u will increase the probability that II-type data can't be served in time, and a proper value of T_u can achieve optimal quality for live streaming. In our experiments, we get a serial of CDF curves of startup delay and chunk arrival ratio with different T_u, and then analyze the impact of T_u on system performance. Figure 6 shows these CDF curves, where the server upload bandwidth is configured to be 10Mbps.

Figure 6 shows that the size of urgent buffer T_u has obvious impact on system performance. When T_u changes from 2 seconds to 4 seconds, startup delay observably decreases about 6 seconds, and chunk arrival ratio also increases clearly. But when T_u is set to be 1 second, the performance of startup delay and chunk arrival ratio drops down. The results indicate that EUE achieve optimal performance when T_u is set to be 2 seconds.

5.2.3 The impact of flash crowd

In EUE principle, the new joining peers download I-type data mostly from servers, but servers are inclined to serve II-type and III-type data, so EUE principle implicitly implements access control. Although this mechanism may prolong the startup delay of new joining peers, it can prevent severe influence on those peers that are watching program.

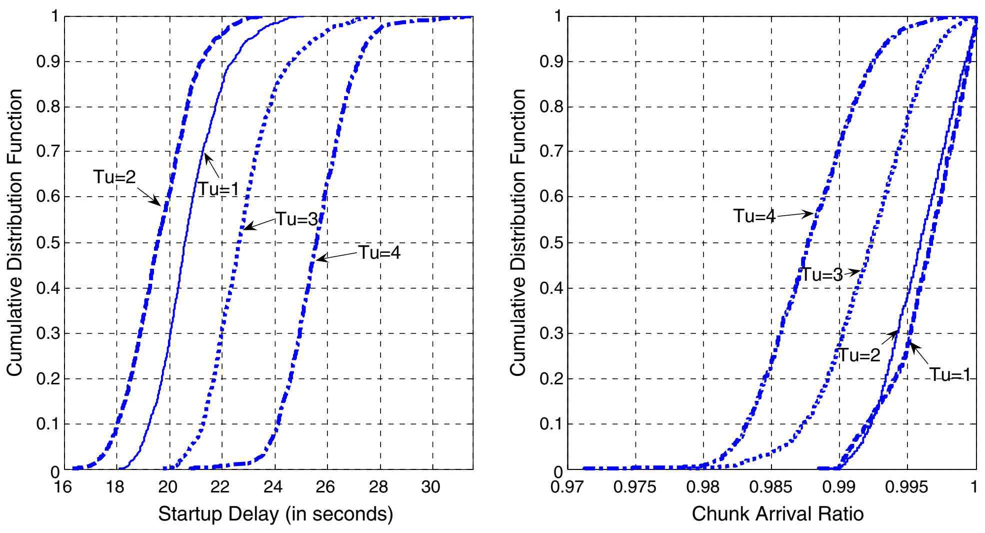

Fig. 6 The impact of T_u on system performance

To analyze the effect of EUE on relieving the impact of flash crowd, we design an experiment as follows. First, 1500 peers are added into the system gradually, and then 500 peers participate in system with the rate of 10 peers every second after system is in stable state. Finally, system performance under different states is compared. Their CDF curves of startup delay and chunk arrival ratio are shown in Fig. 7, where server bandwidth is 10Mbps.

These curves in Fig. 7 show that EUE can effectively tackle flash crowd and those peers that burst into system during a short time have negligible influence on chunk arrival ratio despite their startup delay increase significantly, because servers control the amount of resources allocated to these new comers. In contrast, system performance in TSM, both startup delay and chunk arrival ratio, are affected severely.

6 Conclusions

In order to promote user's QoS of live streaming systems and reduce their maintenance costs, CDN-P2P hybrid architecture has been considered to be an excellent choice for live streaming systems. But how to design resource scheduling mechanism to provide distinct user's demands with differentiated service is still not addressed. In this paper, we propose EUE principle from the view of resources provision and consumption, and present peer resource request mechanism, server resource allocation mechanism, and a corresponding selfish node defense scheme to efficiently assign system resources. Finally, we assess the proposed mechanisms by simulation experiment, and the results show that these mechanisms can reduce startup delay and increase

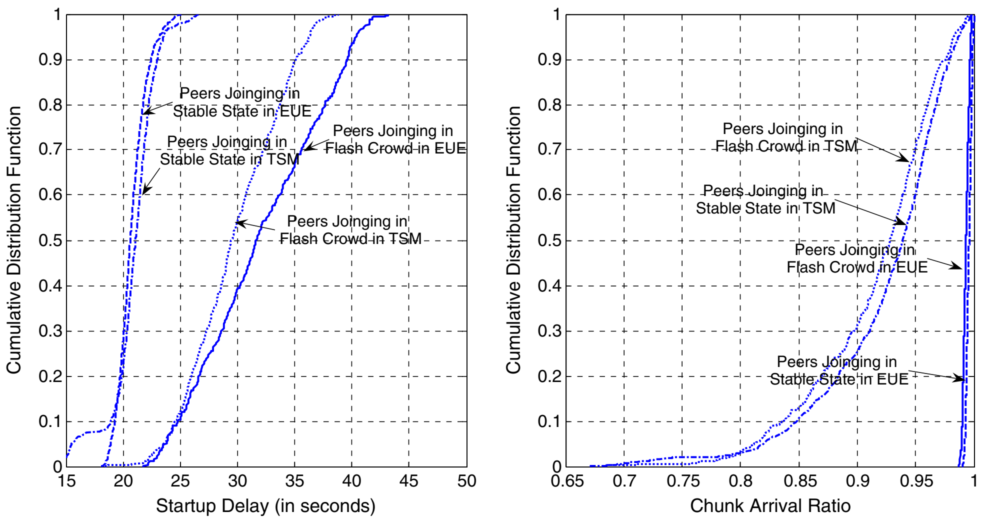

Fig. 7 The impact of flash crowd

chunk arrival ratio remarkably, and effectively improve user's QoS and system performance.

Acknowledgements This work is supported by the National Basic Research Program of China under Grant No. 2012CB315806, the National Natural Science Foundation of China under Grant No. 90304016 and No.61103225, Jiangsu Province Natural Science Foundation of China under Grant No. BK2010133, and LMNT PLAUST. We are very thankful to Lihua Song, Han Wang and Yiting Duan-Mu, who gave helpful advice for our paper's revision.

References

1. Multimedia Research Group Inc., http://www.mrgco.com/iptv/gf1210.html
2. Skevik KA, Goebel V, Plagemann T (2004) Design of a hybrid CDN, In: 2nd International Workshop on Multimedia Interactive Protocols and Systems, Grenoble, France, pp. 206–217
3. Liu Y, Hao Y, Zhu G et al (2008) Peer-assisted content delivery network for live streaming: Architecture and practice, In: International Conference on Networking, Architecture, and Storage, Chongqing, China, pp. 149–150
4. Li B, Xie S, Qu Y et al (2008) Inside the new coolstreaming: principles, measurements and performance implications, In: IEEE INFOCOM'08, Phoenix, AZ, USA, pp. 1031–1039
5. Huang Y, Fu TZJ, Chiu DM et al (2008) Challenges, design and analysis of a large-scale P2P-VoD system, In: ACM SIGCOMM'08, Seattle, Washington, USA, pp. 375–388
6. Zhang X, Liu J, Li B et al (2005) CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming, In: IEEE INFOCOM'05, Miami, USA, pp. 2102–2111
7. Liao X, Jin H, Liu Y et al (2006) AnySee: Peer-to-peer live streaming, In: IEEE INFOCOM'06, Barcelona, Spain, pp. 1–10
8. Wang M, Li B (2007) R^2: Random push with random network coding in live peer-to-peer streaming. IEEE J Sel Area Comm 25 (9):1655–1666
9. Nguyen AT, Li B, Eliassen F (2010) Chameleon: Adaptive peer-to-peer streaming with network coding, In: IEEE INFOCOM'10, San Diego, CA, USA, pp. 1–9
10. Zhou Y, Chiu DM, Lui JCS (2011) A simple model for chunk-scheduling strategies in P2P streaming. IEEE/ACM Transactions on Networking 19(1):42–54
11. Yang Y, Chow ALH, Golubchik L et al (2010) Improving QoS in bitTorrent-like VoD systems, In: IEEE INFOCOM'10, San Diego, CA, USA, pp. 2061–2069
12. Peer-to-Peer streaming simulator, http://media.cs.tsinghua.edu.cn/~zhangm/download/
13. Magharei N, Rejaie R (2009) PRIME: Peer-to-peer receiver-driven mesh-based streaming. IEEE/ACM Transactions on networking 17 (4):1052–1065
14. Wu D, Liang C, Liu Y et al (2009) View-upload decoupling: A redesign of multi-channel P2P video systems, In: IEEE INFOCOM'09, Rio de Janeiro, Brazil, pp. 2726–2730
15. Bonald T, Massoulie L, Mathieu F et al (2008) Epidemic live streaming: Optimal performance trade-offs, In: Proceedings of ACM SIGMETRICS, Annapolis, USA, pp. 325–336
16. Massoulie L, Twigg A, Gkantsidis C et al (2007) Randomized decentralized broadcasting algorithms, In: IEEE INFOCOM'07, Anchorage, Alaska, USA, pp. 1073–1081
17. da Silva APC, Leonardi E, Mellia M et al (2008) A bandwidth-aware scheduling strategy for P2P-TV systems, In: Eighth International Conference on Peer-to-Peer Computing, pp. 279–288
18. Liu Y (2010) Delay bounds of chunk-based peer-to-peer video streaming. IEEE/ACM Transactions on Networking 18(4):2726–2730
19. Abeni L, Kiraly C, Cigno RL (2009) On the optimal scheduling of streaming applications in unstructured meshes, In: Proceedings of Networking, pp. 117–130
20. Fortuna R, Leonardi E, Mellia M et al (2010) QoE in pull based P2P-TV systems: Overlay topology design tradeoffs, In: Tenth International Conference on Peer-to-Peer Computing, pp. 1–10
21. Liu S, Shen RZ, Jiang W et al (2008) Performance bounds for peer-assisted live streaming, In: Proceedings of ACM SIGMETRICS, Annapolis, USA, pp. 313–324
22. Kumar R, Liu Y, Ross K (2007) Stochastic fluid theory for P2P streaming systems, In: IEEE INFOCOM'07, Anchorage, Alaska, USA, pp. 919–927

Chao Hu received the M.S. degree from PLA University of Science and Technology, Nanjing, China, in 2008. He is currently a Ph.D. student at PLA University of Science and Technology. His research interests include peer-to-peer streaming, distributed computing. His mailing address is Department of Computer Engineering, PLA University of Science and Technology, Baixia District, Nanjing 210007, China. His email address is huchaonj@126.com

Ming Chen received the Ph.D. degree from Nanjing Institute of Communication Engineering, China, in 1991. He is currently a professor in the Department of Computer Science and Engineering at PLA University of Science and Technology, Nanjing, China. He held visiting position at Columbia University in 1999. He has published extensively in network architecture, network measurement and monitor, performance evaluation, distributed computing. His mailing address is Department of Computer Engineering, PLA University of Science and Technology, Baixia District, Nanjing 210007, China. His email address is mingchennj@163.com

Changyou Xing received the Ph.D. degree from PLA University of Science and Technology, Nanjing, China, in 2009. His research interests include peer-to-peer multimedia communications, network modeling. His mailing address is Department of Computer Science and Engineering, PLA University of Science and Technology, Baixia District, Nanjing 210007, China. His email address is changyouxing@126.com

Bo Xu received the Ph.D. degree from PLA University of Science and Technology, Nanjing, China, in 2011. His research interests include peer-to-peer measurement, network management. His mailing address is Department of Computer Science and Engineering, PLA University of Science and Technology, Baixia District, Nanjing 210007, China. His email address is xubo820@163.com