A Novel Scheme for Improving Quality of Service of Live Streaming

Guomin Zhang, Chao Hu and Changyou Xing
Department of Network Engineering, College of Command Information System
PLA Univ. of Sci. and Tech. Nanjing, China
E-mail: zhang_gmwn@163.com

Na Wang
Department of Electronic Technology, College of Communication Engineering
PLA Univ. of Sci. and Tech. Nanjing, China

Xianglin Wei
3. PLA University of Science and Technology, Nanjing, China
E-mail: wei_xianglin@163.com

Keywords: live streaming system, peer-to-peer network, quality of service, queuing model

Received: July 16, 2015

P2P live streaming system is one of the most popular Internet applications which developed rapidly in the past decade. However, some common problems, such as long startup delay and unsmooth playback, seriously restrict user's experience on live streaming. In this paper, we propose a novel but simple scheme, namely guarantee mechanism of contingency resource (GMCR), which can improve the quality of service (QoS) of live streaming by deploying a contingency server to provide contingency service for those chunks whose playback deadlines are urgent. Then we establish a queuing model to analyze the quantitative relation between the amount of contingency server resources and the level of user's QoS. Finally, we simulate our scheme in a P2P live streaming simulation platform, and obtain the optimal value of some critical parameters. The results of theoretical analysis and simulation experiment present the feasibility and validity of GMCR scheme.

Povzetek: Predstavljen je nov mehanizem za bolj kvalitetno predvajanje video posnetkov v živo preko spleta.

1 Introduction

Recently, transmitting TV programs over the Internet has become an increasingly popular type of network application. This type of application provides users with abundant, convenient, highly interactive multimedia service, and has engendered a large-scale industry [1]. Live streaming is such a type of application that delivers live program over the Internet, and involves a camera for the media, an encoder to digitize the content, a media publisher, and a content delivery network to distribute and deliver the content [2].

According to the transmission mode in the Internet, the content delivery network of live streaming can be divided into client/server (C/S) paradigm and peer-to-peer (P2P) paradigm. C/S paradigm provides live program from servers entirely, but this transmission mode brings many problems, such as poor scalability and high costs, which makes it difficult to implement large-scale deployment. On the contrary, P2P paradigm exploits the idle resources in end users effectively, who can share and exchange their video chunks, thus improving the operation efficiency and decreasing the costs. There exist many live streaming systems, such as Coolstreaming [6], PPLive, PRIME [13]etc. However, despite P2P can meet the demands of file sharing application, it can't provide QoS guarantee for time-sensitive and bandwidth-sensitive applications, e.g. live streaming. Bo Li et al. measured Coolstreaming and discovered that only about 95% of the chunks could reach user's buffer before being played [5], while Yan Huang analyzed the data collected from PPLive and found the buffering time of almost 20% of the users occupied more than 80% of the total time [6]. Due to these missing chunks, the screen will be frozen, and the media players have to wait until these chunks arrive at the user's buffer, which seriously degrades the quality of experience. An intuitive solution is to increase the amount of streaming servers to cut down the loss rate of the chunks, but in another aspect, too many servers deployed in the Internet will raise the costs and result in the waste of resources.

Therefore, studying the relation between the amount of server resources and user's QoS level, and finding out a scheme to achieve the balance between them have great theoretical significance and practical importance. However, the instinct of IP network is *best effort*, and it's difficult to provide rigorous QoS for users, but the statistical analysis still has important reference to our research. In this paper, from the perspective of improving the QoS of live streaming, we propose a novel but simple scheme, namely guarantee mechanism of contingency resource (GMCR for short), which deploys a contingency server to provide necessary countermeasure to those potential missing chunks to promote user's QoS. Subsequently, we establish a queuing model to analyze the quantitative relation between the resources of contingency server and user's QoS.

The rest of this paper is organized as follows. Related work is introduced in Section 2. GMCR is proposed in Section 3. The quantitative relation of contingency server resources and user's QoS is analyzed by queuing model in Section 4. The performance of GMCR and the results of theoretical model are evaluated by simulation experiment in Section 5. Finally, our work is concluded in Section 6.

2 Related work

Because P2P can alleviate the pressure of the streaming servers and make use of the peers' resources, P2P technology was first introduced to a practical live streaming system in [6]. A data-driven based overlay network, namely DONet, was constructed to implement better transmission and dissemination of the live streaming data. Moreover, AnySee and PRIME [13] also used P2P for the deployment and operation of live streaming systems, and the use of the peers' resources effectively decreased the cost of the servers. However, the intrinsic characteristics of the peers of P2P networks, including dynamic and peer churn, as well as the impact of firewall and network address translation, make it impossible to provide high QoS for the users if a live streaming system entirely relies on the resources from the peers. In [3], CDN and P2P hybrid architecture was proposed to disseminate video streaming. Under this circumstance, there are more server resources, which can lead to better QoS. In [4], a peer-assisted content delivery network was proposed, in which there are two layers, and CDN distribution layer lies in the core network while P2P distribution layer is deployed in the access network. In [14], a radically different cross-channel P2P streaming framework, namely View-Upload Decoupling (VUD), was proposed, which decouples the peer downloading from uploading, bringing stability to multi-channel systems and enabling cross-channel resource sharing to make sure the resource provision is sufficient for user demand in each channel.

Besides, improving resource utility is another way to improve user experience. In [11], a mixed strategy to schedule video chunk was proposed, and it remarkably increased the rate of data arriving at the users' buffer in time. In [12], Yan Yang et al. introduced a deadline-aware scheduling approach, which avoided the waste of resources to a certain extent by considering the data request deadline. In [9], random network coding was employed to P2P live streaming, and it polished up the system performance. In [10], a new P2P streaming algorithm that incorporated the network coding seamlessly with the scalable video coding was designed, and the experiments demonstrated the feasibility and better performance of the approach.

Actual video streams carry highly organized information, part of which is more important than others, and with high variability in the generated bitrate. Chunk loss probability and delivery delay provide therefore only a partial view of the actual performance of a P2P-TV system, the user Quality of Experience (QoE) being the paramount index [15]. In the multimedia and signal processing communities, indeed, the evaluation of the QoE is considered mandatory, see [16], [17] for notable examples. In [15], a realistic simulative model of the system was proposed, which represented the effects of access bandwidth heterogeneity, latencies, peculiar characteristics of the video, while still guaranteeing good scalability properties. Otherwise, a new latency/ bandwidth-aware overlay topology design strategy was proposed, which improved application layer performance while reducing the underlying transport network stress. Reference [15] investigated the impact of chunk scheduling algorithms that explicitly exploit properties of encoded video.

In [18], Hu et al. studied the chunk dissemination of P2P live streaming, and introduce a discrete and slotted mathematical model to analyze chunk selection algorithms, including rarest first algorithm and greedy algorithm. Moreover, Xing et al. presented a performance metric to evaluate chunk selection algorithms, as well as the optimization function for the exploration of chunk dissemination strategies. Reference [19] pointed out the causes of poor performance of these algorithms, and propose a service request randomization mechanism to promote the use of peer resources, which can prevent chunk requests from rendezvous on a few of peers. Simultaneously, they employ weight assignment strategies to avoid excessive requests for rare chunks. Besides, an enhanced model was presented, which adds node degree constraint.

Simultaneously, analyzing P2P live streaming with mathematical model is also a hot spot. In [11], a discrete and slotted model was adopted to study the chunk selection strategy of P2P live streaming. Kumar et al employed stochastic fluid theory to model P2P streaming systems, and exposed the fundamental characteristics and limitations in [20].

When some popular programs start, many users will access this channel during a short time, and if these requests couldn't be handled appropriately, severe performance problem will emerge. Another crucial issue is peer churn, and robust live streaming systems must have the capability against peer dynamics. Practical chunk scheduling mechanisms must be able to deal with these issues. In [21], a radically different cross-channel P2P streaming framework, namely View-Upload Decoupling (VUD), was proposed, which decouples the peer downloading from uploading, bringing stability to multi-channel systems and enabling cross-channel resource sharing to make sure the resource provision is sufficient for user demand in each channel. Kumar et al employed stochastic fluid theory to model P2P streaming systems, and exposed the fundamental characteristics and limitations in [22]. Liu et al. theoretically studied chunk-based P2P video streaming in [23], and showed the delay bound to distribute video chunks to all peers. Furthermore, a conceptual snowball streaming algorithm was proposed to approach the minimum delay bound in dynamic P2P network environment. In [25], the authors presented a novel metric, called the Content Propagation Metric (CPM), to quantitatively evaluate the marginal benefit of available bandwidth, and CPM could guide a global allocation of bandwidth to maximize the aggregate download bandwidth of consumers.

3 Guarantee Mechanism of Contingency Resource

3.1 Basic idea

In P2P live streaming system, source server codes the live television signal, and periodically generates new video chunks, and then distributes these chunks to peers in P2P network. Subsequently, peers share and exchange their possessed chunks to take charge of the partial uploading assignment for source server, which enables source server without powerful upload capacity to provide live service for a great number of users, and improve the scalability of live streaming system.

However, in contrast to the dedicated server, the upload capacity of ordinary peers is limited. Especially, there are plenty of peers locating behind firewall or network address translation, and those devices restrict the resources usage of P2P network, where network resources includes processing resources and bandwidth resources etc., but the performance bottleneck of live streaming system generally lies on uplink bandwidth (In order to prevent terminological confusion, we don’t distinguish resource and uplink bandwidth in the following content). Furthermore, streaming application is different from file-sharing application in time sensitivity. If a chunk doesn’t arrive at peer before being played, it is equivalent to chunk miss even though the chunk reaches user buffer afterwards. Consequently, it is necessary to ensure the amount of resources provided by server or peers must be adequate for the resource requirements of peers. Assume a live channel has *n* peers and source server’s uplink bandwidth is u_s, the uplink bandwidth of peer *i* is u_i, and the playback rate is *r*, then the precondition that all peers can watch live program smoothly is:

$$\frac{u_s + \sum_{i=1}^{n} u_i}{nr} \geq 1 \tag{1}$$

For a single peer, it also needs enough resources for continuous playing. Assume the instantaneous bandwidth peer *i* derives from source server is u_{is}, and the value is u_{ij} from peer j, where $u_{ii}=0$, and then the instantaneous total bandwidth that peer *i* receives is $u_{is} + \sum_{j=1}^{n} u_{ij}$. For each peer, it can receive sufficient in instantaneous state when equation (2) is satisfied.

$$\begin{cases} u_{is} + \sum_{j=1}^{n} u_{ij} \geq r,\ for\ all\ i = 1,2,...,n \\ \sum_{i=1}^{n} u_{is} \leq u_s \\ \sum_{i=1}^{n} u_{ij} \leq u_j,\ for\ all\ j = 1,2,...,n \end{cases} \tag{2}$$

Equation (1) ensures that there are enough resources for users, while it also makes equation (2) have feasible solutions of resource allocation. However, because of the uncertainty of resource scheduling and delay sensitivity of live streaming, it’s difficult to guarantee that every peer can receive sufficient resources to download video chunks for smooth playing every time despite the total resources exceeds users’ requirements from the macroscopic angle.

Every video chunk stored in peer buffer has a deadline apart from being played. Assume the playback deadline is t_p, and this deadline will decrease with the time’s lapse. When chunk’s deadline is less than a certain threshold, and the chunk hasn’t received yet, it is in the risk of missing. Suppose the threshold value is T_u, and if the playback deadline of unpossessed chunk satisfies $t_p \leqslant T_u$, then this type of chunk is called ***urgent chunk***, and the other chunk is ***non-urgent chunk***.

In contrast with those non-urgent chunks, urgent chunk should be served firstly, because the loss probability is much larger than that of non-urgent chunk. Nevertheless, the program progresses of different users are diverse in practical live streaming system, and the resource owners can’t distinguish which chunk is nearer to be played, and they can’t easily find out the more urgent chunk requests and serve them. Consequently, if there is a dedicated server for those urgent chunks to provide contingency service, it’s likely to reduce the loss probability with fewer resources, and improve user’s QoS.

3.2 The model of Guarantee Mechanism of Contingency Resource

Based on the above idea, we propose a scheme, namely guarantee mechanism of contingency resource (GMCR), to promote system performance. GMCR is a scheme to provide contingency service for urgent chunks in order to make sure they can arrive at user’s buffer in time. In practical P2P live streaming system, we can deploy a server that accomplishes GMCR to serve those urgent chunks, and this server is called ***contingency server***. It is a viable solution to the improvement of live streaming QoS to resort to the coordination of contingency server and P2P network.

In order to implement GMCR, we need to adjust chunk scheduling mechanism appropriately. Firstly, peer should partition all the unreceived chunks into urgent chunks and non-urgent chunks according to the value of t_p and T_u. For non-urgent chunks, peer requests them from other peers or source server based on P2P paradigm. Once a non-urgent chunk becomes urgent due to the decrease of t_p, the peer sends chunk request to contingency server immediately, and contingency server responses to this request promptly to provide this urgent chunk. Figure 1 shows the model of GMCR.

In Figure 1, source server and peers constitute the typical P2P live streaming system, and they share and exchange video chunks with each other, while contingency server is dedicated to provide contingency service for urgent chunk request. Peers ought to decide the resource requested object based on the deadline of chunk.

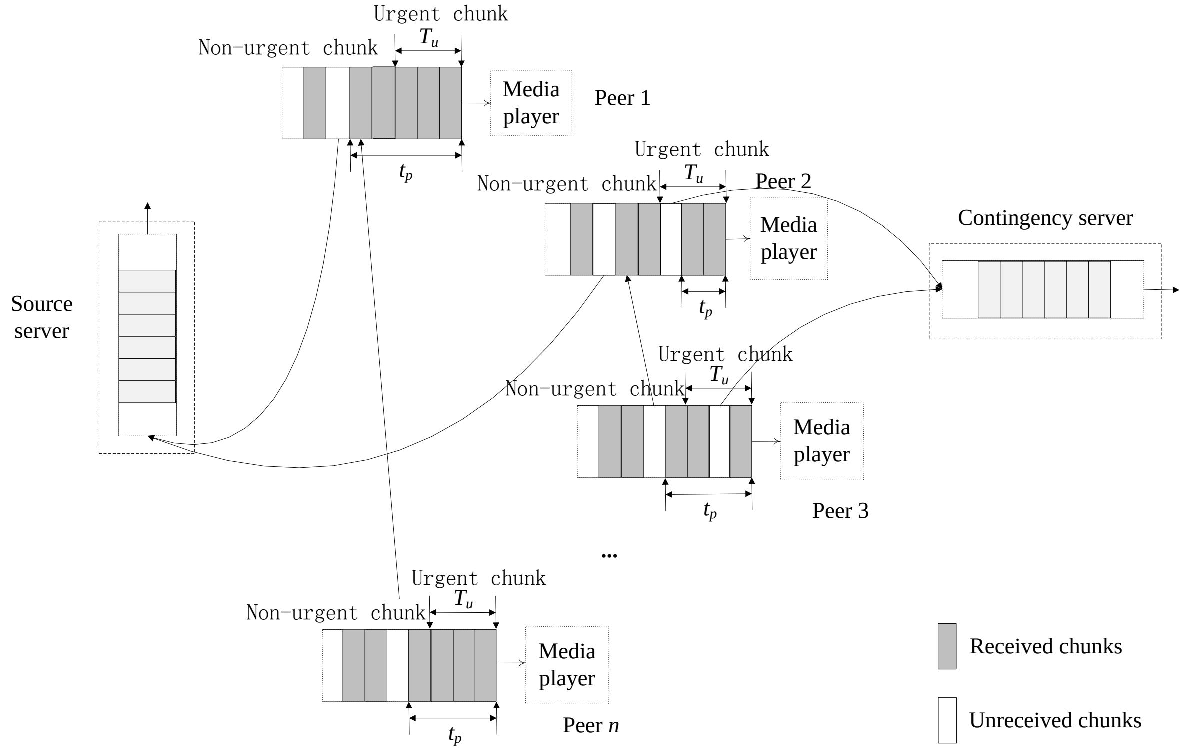

Figure 1: The model of GMCR.

4 Queuing model analysis of Guarantee Mechanism of Contingency Resource

In this section, we analyze the GMCR model quantitatively with queuing theory to discover the relation between the resource amount of contingency server and users' QoS.

4.1 Model and notations

According to GMCR, if the playback deadline of a chunk in peer's buffer t_p is less than T_u, the peer will send request to contingency server immediately to prevent chunk missing. Subsequently, contingency server is going to insert these urgent chunk requests into service queue, and provide contingency service for them to keep the chunk from missing caused by time out. Figure 2 shows the queuing model depicting the above work flow.

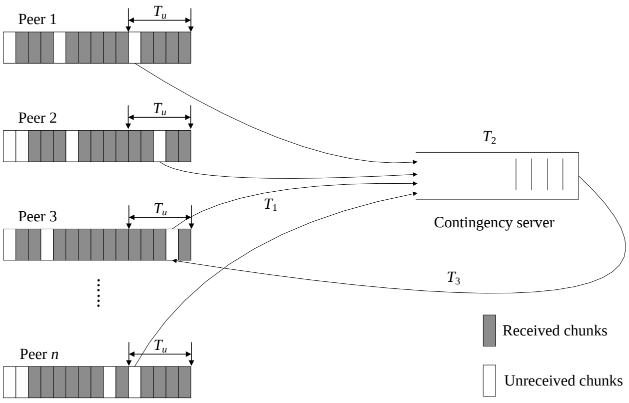

Figure 2: The queuing model for contingency.

To describe the model, we define the system parameters and notations in Table I.

Table 1: Notation and definition of GMCR queuing model.

Notation	Definition
T_u	Playback deadline threshold of urgent chunks
t_p	Playback deadline of chunk away from being played
U	Uplink bandwidth of contingency server
N	The number of peers in live streaming system
R	Playback rate of live program
L	Size of each chunk in live streaming
B	Queue length of contingency server to store urgent chunk requests
a	The probability of chunk that doesn't arrive at peer buffer when its playback deadline reduces to T_u
T_1	The delay of peer sending urgent chunk request to contingency server
T_2	The queuing delay of urgent chunk request in contingency server
T_3	The delay of contingency server sending urgent chunk to peer

4.2 Performance analysis

To analyze the model, we divide time into slots firstly, and the size of slot is equal to the period that source server generates a new video chunk, so the size of each slot is

$$T = L / R \tag{3}$$

When a slot passes, at most one non-urgent chunk will turn into urgent chunk in every peer, and it is going to send an urgent chunk request to contingency server. As defined, *a* is the probability of the chunk that doesn't arrive at peer buffer when deadline reduces to T_u, which means the probability every peer will send request in each slot is *a*. Because the peer's number is *N* in system, the probability that contingency server will receive *k* urgent chunk requests in a slot is

$$a_k = \binom{N}{k} a^k b^{N-k} \tag{4}$$

Where *b*=1-*a*.

On the other hand, the most amount of urgent chunk that contingency server can upload in a slot is

$$M = \left\lfloor \frac{U}{R} \right\rfloor \tag{5}$$

When the arrival ratio of urgent chunk request is less than the capacity of contingency server, the server can handle all the requests in one slot. But if the arrival ratio exceeds the server's capacity, part of the requests have to stay in contingency server's queue.

Assume the state probability that there are *k* urgent chunk requests in contingency server queue to wait for service is s_k, while the queue length is *B*, so all of the state probabilities constitute a state vector as follows.

$$S = [s_0 \; s_1 \; s_2 \ldots s_B] \tag{6}$$

According to the arrival ratio of chunk request in each slot and contingency server's capacity, we can obtain the state transition diagram in Figure 3.

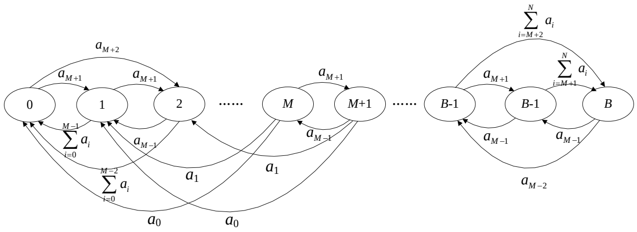

Figure 3: State transition diagram of contingency server's queue.

Therefore, the state transition matrix is denoted as *P*, and the element p_{ij} in the matrix means the probability of the number of urgent chunk request in contingency server's queue transiting from *i*-1 to *j*-1.

$$P = \begin{bmatrix} \sum_{i=0}^{M} a_i & a_{M+1} & a_{M+2} & \ldots & a_N & 0 & \ldots & 0 \\ \sum_{i=0}^{M-1} a_i & a_M & a_{M+1} & \ldots & a_{N-1} & a_N & \ldots & 0 \\ \sum_{i=0}^{M-2} a_i & a_{M-1} & a_M & \ldots & a_{N-2} & a_{N-1} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_0 & a_1 & a_2 & \ldots & a_{N-M} & a_{N-M+1} & \ldots & \sum_{i=B-M}^{N} a_i \\ 0 & a_0 & a_1 & \ldots & a_{N-M-1} & a_{N-M} & \ldots & \sum_{i=B-M-1}^{N} a_i \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & \ldots & \ldots & \ldots & a_M \end{bmatrix} \tag{7}$$

At equilibrium, the input probability is equal to the output probability for every state, so the equilibrium equation is given as

$$SP = S \tag{8}$$

Meanwhile, the queue length of contingency server is *B*, so we can obtain the following condition

$$\sum_{i=0}^{B} s_i = 1 \tag{9}$$

Substitute equation (6) and (7) for equation (8), and combine equation (9), we can compute the state probability of contingency server's queue.

For every urgent chunk, if peer can receive this chunk from contingency server in T_u, this video chunk can be played on schedule, otherwise it will be lost. When a peer wants to get urgent chunk, it must suffer three phase latencies, and they are T_1, T_2, and T_3, respectively. Consequently, the condition that ensures urgent chunk arrives in time is T_1+ T_2+ $T_3 \leqslant T_u$, and the longest waiting time of urgent chunk request in contingency server is T_u- T_1- T_3. Because contingency server can deal with $\left\lfloor \frac{U}{R} \right\rfloor$ requests in one slot, the chunk's number that an urgent chunk request can wait for is given by

$$W_m = \left\lfloor \frac{(T_u - T_1 - T_3)}{T} \times \frac{U}{R} \right\rfloor = \left\lfloor \frac{(T_u - T_1 - T_3)U}{L} \right\rfloor \tag{10}$$

Therefore, urgent chunk request can arrive at peer buffer in time when the number of chunk request in contingency server's queue doesn't exceed W_m, and the loss probability p_{loss} is approximately equal to

$$p_{loss} \approx 1 - \sum_{i=0}^{W_m} s_i \tag{11}$$

For instance, if there are 1,000 peers in live streaming system, and the playback rate is 400kbps, the size of chunk is 64kB, and the arrival ratio of urgent chunk is 5%, while contingency server's uplink bandwidth is 21Mbps, and its buffer can store 1,000 urgent chunk requests, the value of T_u, T_1 and T_3 is 2s, 100ms and 100ms, respectively, then we can figure out the loss probability is 0.015% according to equation (11). In fact, considering the quasi-synchronous characteristic of live streaming, all the operation to offer the contingency service can be implemented in the memory of server, and a server can provide service for more users with lower loss probability.

5 Simulation experiment

To validate the feasibility and potential performance of GMCR, we test this mechanism in a P2P streaming simulator, namely P2PStrmSim [24], and compare GMCR to P2P-only mechanism. The purpose of our simulation is to check the dependability of GMCR queuing model, and we contrast the simulation results to theoretical results.

5.1 Experiment scenario and metrics

P2PStrmSim is an event-driven P2P live streaming simulator, and it can simulate the packet-level data exchange process. All the events, including control packet exchange and data packet transmission etc., are stored in event queue; meanwhile, they are managed and executed by event engine. When an event is executed, the corresponding operation will be called. The system framework of P2PStrmSim is depicted as Figure 4.

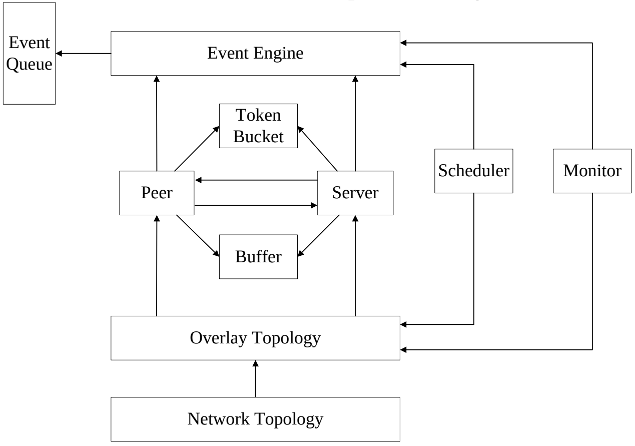

Figure 4: The system framework of P2PStrmSim.

In P2PStrmSim, network topology is based on the measurement results of Internet delay, and it ignores the impact of packet queuing in routers. Peers construct overlay topology on network topology, and every peer has 15 neighbors. The uplink bandwidth of server and peers are implemented by token bucket, and all video chunks are stored in buffer. The function of scheduler is to schedule peers' arrival and departure, while monitor measures the system parameters and performance metrics. Event queue stores all the events of live streaming system generated in the simulation process, including peers' join/leave system, buffer map exchanging between peers to be aware of which chunk other peers have received, as well as sending and reception of the packet. Event engine inserts newly generated event into event queue, and execute the event in the front of event queue. By configuring system parameters briefly, P2PStrmSim can directly simulate the whole process of P2P live streaming. In order to simulate GMCR, we modify the original simulator and append correlative module to implement contingency server. Contingency server takes charge of providing service to urgent chunks, and we introduce three types of delay to denote T_1, T_2 and T_3. In peer module, we add the function of classifying urgent and non-urgent chunks based on T_u, and non-urgent chunks are requested by typical P2P paradigm, while urgent chunk requests are sent to contingency server. The simplicity of developing GMCR also indirectly demonstrates the feasibility of this mechanism.

The scheduler can adjust communication paradigm according to the lowest quality requirement of various paradigm after obtaining performance information. There are mainly two parts in adaptive communication mechanism: network performance awareness module and adaptive paradigm adjustment module. The former is devoted to obtain end to end performance information through measurement and inference technology; and the latter intends to adopt the optimum communication paradigm to fulfill user requirement.

In practical experiment scenario, the uplink bandwidth of source server is 10Mbps, and it generates video chunk with the rate of 400kbps, contingency server can store 1,000 urgent chunk requests. Simultaneously, some peers in the system download the video chunks according to the fixed scheduling mechanism. Moreover, the request window is 30s, and the size of the video chunk is 64kB. All peers access the network by asymmetrical digital subscriber line (ADSL), and the downlink bandwidth exceeds the uplink bandwidth. In order to simulate the heterogeneity of the peer's access bandwidth, we introduce three types of ADSL, whose uplink bandwidths are 1Mbps, 384kbps and 128kbps, respectively, and their proportions are given in Table II.

Table 2: The proportion of the three type of uplink bandwidth in the simulation experiment.

Uplink bandwidth	Proportion in experiment
1 Mbps	0.2
384 kbps	0.45
128 kbps	0.35

We evaluate the performance of GMCR by monitoring the continuity of live program. Once a video chunk doesn't arrive at peer's buffer before being played, it will lead to program pause or screen frozen, so we employ chunk arrival ratio (CAR), which is equal to the ratio of chunks arriving in time and the total chunks, as the metric to evaluate user QoS. Obviously, the higher CAR means users have received higher QoS. In our simulation, all the results are the average value of 10 experiment results tested with different seeds.

5.2 Experiment results and analysis

5.2.1 Performance of Guarantee Mechanism of Contingency Resource

Figure 5 shows the CAR's cumulative distribution function (CDF) of GMCR and P2P based live streaming in different scale channels, where N is the peer's number, the uplink bandwidth of contingency server is 2Mbps, and T_u is 2s.

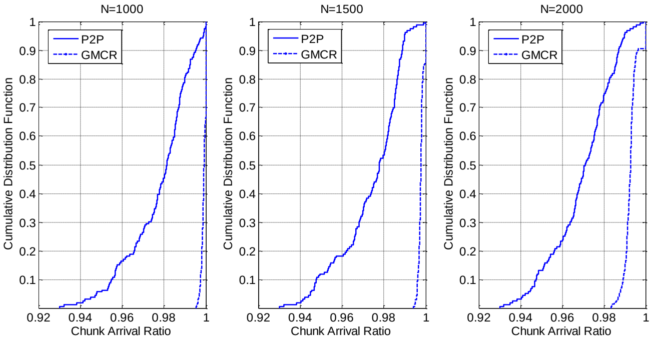

Figure 5: The CDF of CAR in GMCR and P2P based live streaming system.

From Figure 4 we can find GMCR can provide better QoS for users in all different scale channels, and almost all the peers have a chunk arrival ratio above 99%. But in typical P2P paradigm, there are more than 15% of the peers whose loss probabilities exceed 4%. Though the uplink bandwidth of source server is not high, contingency server is deployed in live streaming system and it can provide contingency service to the peers when they have urgent chunk, which greatly reduces the amount of chunk missing and improves CAR. Furthermore, the CARs of GMCR and P2P based system decreases in different degree with the increment of peers. This trend of performance degradation is due to the augment of user demand on the bandwidth resource, and the total amount resource provision of source server and contingency server is invariable.

5.2.2 The impact of T_u

According to the analysis in Section 3, the length of urgent chunk's deadline threshold T_u will affect system performance, because T_u decides the probability of urgent chunk generated in peer buffer and the length of queuing time that these urgent chunk requests can wait in contingency server. If the value of T_u is set to be larger, the probability of non-urgent chunk turning into urgent chunk will increase, and then more urgent chunk requests reach the contingency server, thus augmenting its burden. But if the value of T_u is set to be too small, the chunk request that can stay in contingency server will shorten, thus also increasing the probability of urgent chunks that cannot be played in time. Hence, there exists an appropriate value of T_u that could achieve the optimal system performance. In our experiment, we adjust T_u to get a serial of CDF curves of chunk arrival ratio. Figure 6 shows the CDF of chunk arrival ratio with different T_u.

From the movement trend of Figure 6, we can analyze the rough impact of T_u. Obviously, the curves shows that the value of T_u has significant influence on system performance. When T_u changes from 4 seconds to 2 seconds, chunk arrival ratio increases clearly. But when T_u is set to be 1 second, chunk arrival ratio drops down. This result indicates that GMCR can provide fine contingency service and avoid too many urgent chunk requests when T_u is set to be about 2 seconds.

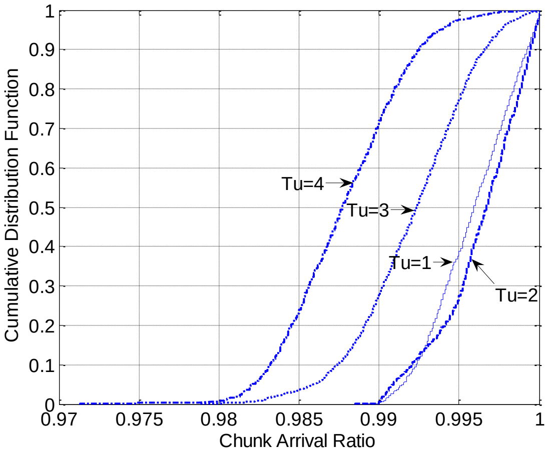

Figure 6: The influence of T_u on system performance.

5.2.3 The impact of contingency server resource

Subsequently, we analyze how to deploy contingency server resource in GMCR based live streaming. Theoretically, many urgent chunk requests are unable to be served without sufficient resource, and the user QoS will deteriorate. But if too much resource of contingency server is deployed in the system, it will cause the waste of resource. Therefore, it is necessary to supply appropriate contingency server resource according to the practical requirement. Figure 7 shows the movement curves of loss probability in contingency server under different scale channel when the uplink bandwidth of contingency server changes from 1.95Mbps to 2.1Mbps. At the same time, Figure 7 also shows the theoretical value to validate the dependability of GMCR queuing model established in Section 4.

According to Figure 7, we can obtain three conclusions. Firstly, in three different scale channels, the theoretical results are very close to the experimental results, which demonstrates GMCR queuing model can describe the quantitative relation between contingency server resource and user QoS, and this queuing model has great dependability. Secondly, when contingency server has insufficient resource, the loss probability is very high, but if the amount of resource exceeds the demand of urgent chunk request, user QoS will improve significantly. Hence, we should deploy a few more contingency server resources. Thirdly, the requirement of contingency server resources will increase with the augment of peer's number, because this situation will incur the insufficiency of resource provision and result in the increment of urgent chunk request, and more contingency server resources are needed for contingency service. But the requirement increment of contingency server resource is not notable, where the resource amount only increases from 1.95Mbps to 2.1Mbps.

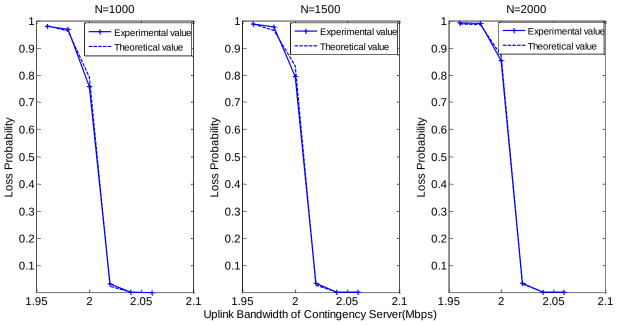

Figure 7: The loss probability of theoretical value and experimental value in contingency server under different scale channel.

6 Conclusion

In this paper, we propose a novel but simple scheme, namely guarantee mechanism of contingency resource, which greatly improves the QoS of live streaming system by deploying contingency server to provide service for urgent chunk request in time. We also establish GMCR queuing model to analyze the quantitative relation between the amount of contingency server resource and user QoS. Finally, we simulate our scheme in simulation experiment, and obtain some conclusions. The experimental results and the theoretical analysis show the dependability and validity of GMCR and this queuing model. Our work sheds light of a new approach for the QoS elevation of live streaming system. In the future, we will focus on the technological approach to construct mathematical model on the different requirement of system resource in distinct phase to improve the QoS of live streaming system.

Acknowledgement

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. 61402521, 61070173 and No. 61103225, and Jiangsu Province Natural Science Foundation of China under Grant No. BK20140070 and BK20140068.

References

[1] Multimedia Research Group Inc. [EB/OL]. Available: http://www.mrgco.com/iptv/ gf1210.html.

[2] Live streaming. Available [EB/OL]: http://en.wikipedia.org/wiki/Streaming_media.

[3] Karl Skevik, Vera Goebel, Thomas Plagemann. Design of a hybrid CDN [C]. Second International Workshop on Multimedia Interactive Protocols and Systems, Grenoble, France, 2004: 206-217.

[4] Yu Liu, Yin Hao, Guangxi Zhu, et al. Peer-assisted content delivery network for live streaming: architecture and practice [C]. International Conference on Networking, Architecture, and Storage, Chongqing, China, 2008: 149-150.

[5] Bo Li, Susu Xie, Yang Qu, et al. Inside the new coolstreaming: principles, measurements and performance implications [C]. INFOCOM, Phoenix, USA, 2008: 1031-1039.

[6] Yan Huang, Tom Z. J. Fu, Dah-Ming Chiu, et al. Challenges, design and analysis of a large-scale P2P-VoD system [C]. ACM SIGCOMM, Seattle, Washington, USA, 2008: 375-388.

[7] Xinyan Zhang, Jiangchuan Liu, Bo Li, et al. CoolStreaming/DONet: a data-driven overlay network for efficient live media streaming [C]. INFOCOM, Miami, USA, 2005: 2102-2111.

[8] Xiaoqun Yuan, Geyong Min, Yi Ding, et al. Adaptive resource management for P2P live streaming systems[J], Future Generation Computer Systems, Vol.29, No.6, 2013.08: 1573 - 1582.

[9] Mea Wang, Baochun Li. R2: random push with random network coding in live peer-to-peer streaming [J]. Journal on Selected Areas in Communications, 2007, 25(9): 1655-1666.

[10] Anh Tuan Nguyen, Baochun Li, Frank Eliassen. Chameleon: adaptive peer-to-peer streaming with network coding [C]. IEEE INFOCOM, San Diego, CA, USA, 2010: 1-9.

[11] Yipeng Zhou, Dah-Ming Chiu, John C.S. Lui. A simple model for chunk-scheduling strategies in P2P streaming [J]. IEEE/ACM Transactions on Networking, 2011, 19(1): 42-54.

[12] Yan Yang, Alix L.H. Chow, Leana Golubchik, et al. Improving QoS in bitTorrent-like VoD systems [C]. INFOCOM, San Diego, CA, USA, 2010: 2061-2069.

[13] Nazanin Magharei, Reza Rejaie. PRIME: peer-to-peer receiver-driven mesh-based streaming [J]. Transactions on networking, 2009, 17(4): 1052-1065.

[14] Di Wu, Chao Liang, Yong, Liu, et al. View-Upload Decoupling: A Redesign of Multi-Channel P2P Video Systems [C]. INFOCOM, Janeiro, Brazil, 2009: 2726-2730.

[15] Fortuna, R., Leonardi, E., Mellia, M., Meo, M., Traverso S. QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs[C]. IEEE P2P 2010

[16] Z. Shen and R. Zimmermann, ISP-friendly peer selection in p2p networks[C]. in ACM Multimedia, Beijing, China, October 2009.

[17] E. Setton, J. Noh, and B. Girod, Low latency video streaming over peer-to-peer networks[C]. in IEEE ICME, Toronto, Canada, July 2006.

[18] HU Chao, CHEN Ming, XING Changyou. Towards Efficient Video Chunk Dissemination in Peer-to-Peer Live Streaming[J]. Computer Networks, Vol. 57, issue 15, pp.3009-3024, 2013.

[19] XING Changyou, CHEN Ming, HU Chao. Capacity aware Scalable Video Coding in P2P on Demand Streaming Systems[J]. KSII Transactions on Internet and Information Systems, Vol. 7, issue 9, pp. 2268-2283, 2013.

[20] Rakesh Kumar, Yong Liu, Keith Ross. Stochastic Fluid Theory for P2P Streaming Systems. IEEE INFOCOM, Anchorage, Alaska, USA, 2007: 919-927.

[21] R. S. Peterson, B. Wong, E. G. Sirer, A content propagation metric for efficient content distribution[C], in: ACM SIGCOMM 2011, New York, NY, USA, 2011, pp. 326-337.

[22] A. P. C. da Silva, E. Leonardi, M. Mellia, M. Meo, S. Traverso, A bandwidth-aware scheduling strategy for P2P-TV systems[C], in: 8th International Conference on Peer-to-Peer Computing, Aachen, Germany, 2008, pp. 279-288.

[23] Z. Liu, C. Wu, B. Li, S. Zhao, UUSee: large-scale operational on-demand streaming with random network coding[C], in: IEEE INFOCOM 2010, San Diego, CA, USA, 2010, pp. 1-9.

[24] Peer-to-Peer Streaming Simulator. http://media.cs.tsinghua.edu.cn/~zhangm/download/. 2012

[25] R. S. Peterson, B. Wong, E. G. Sirer, A content propagation metric for efficient content distribution, in: ACM SIGCOMM 2011, New York, NY, USA, 2011, pp. 326-337.