Provide VoD Service in Peer-to-Peer Network using Offset Ranking and Epidemic Diffusion

Xing Chen, Qing Yang
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Science
{xchen, qyang}@nlpr.ia.ac.cn

***Abstract*—The Peer-to-Peer solution provides a service model that is attractive for Video-on-demand (VOD). The key idea of using P2P technology is to aggregate storage and bandwidths capacity of peers to alleviate workload on the source server. Compared with P2P live streaming, it is much more challenge for VOD due to peers' asynchronous request, different interested portion and support for VCR-like operation. In this paper, we utilize two-scale list of peers, which is composed of R-peers and S-peers, to locate collaborator and support VCR operation. Demand-driven chunk swarming is proposed to exchange data with ragged collaborators in the VoD system. Simulation results show that the source sever load is significantly reduced, playback continuity is guaranteed and our method supports VCR operation quite well.**

I. Introduction

With the increase of the link capacity offered to Internet end users, media services are rapidly gaining popularity. For example, YouTube, with about 100 million video views and 65,000 video uploads per day, accounts for approximately 60% of the videos watched on the Internet [1]. However, providing media service to a large number of users requires a significant amount of bandwidth, which crucially becomes the scalability bottleneck.

Recently, Peer-to-Peer (P2P) network have been successfully deployed for file share and live media streaming. Peers act as both clients and servers, and contribute resource to alleviate the workload imposed on the server and distribute the bandwidth requirements across the network by share peers' upload bandwidth, P2P solution is efficient in data distribution because it significantly decreases the cost of server and increases the scalability. Ideally, with altruism behavior, P2P system can support arbitrary number of users.

Providing media service in P2P fashion is a growing trend, P2P live streaming systems such as ESM [2], Coolstreaming [3], PPlive [4] have been deployed to a large number of users. Generally, these systems deliver blocks of the video stream to peers who assemble these blocks into video streams and then serve them to the media decoder. Collaborative member management and chuck schedule methods have been studied in order to meet real-time requirement, users' quality-of-experience (QoE), and adaptation to churn in P2P streaming application.

Inspired by the success delivery of static files (*e.g.*, BitTorrent) and live streaming (*e.g*, coolstreaming), can VoD system alleviate chunk swarming technology to diffuse the stored media content? Before we address this problem, we compare VoD with live streaming carefully. They share several common challenges including the sequential playback demands of large media objects, the heterogeneous users, and the dynamic churn in P2P network. Although there are points of similarity between them, some notable difference exists. (1) Peers who join the live streaming at any time don't retrieve earlier part of the streaming (*i.e.* joining an existing stream). Peers in the VoD service need to receive the entire stored media from the beginning (*i.e.* starting a new stream). (2) Peers in a live streaming scenario have *shared temporal content focus*, while they have *greater temporal diversity* of requests [5] in VoD system. (3)User interactivity such as random seeks, rewind, and fast forward should be supported in a VoD system.

In a P2P media swarming system, media content is divided into small size data blocks, the source server disperses the data blocks to different peers. Peers download the missing blocks from their neighbor peers or from the source server directly. To fully utilize peers' upload bandwidth, the system should preserve data diversification among peers so that there are available blocks to exchange. The fact that different users may be watching different part of the video at any time in VoD system can greatly reduce the efficiency of content swarming protocol, and swarming protocol adopted in P2P live streaming system cannot be applied in P2P VoD system directly.

In this paper, we address the asynchronous problem in VoD system with playback time offset model. Peers who join the system randomly are ranked by their playback offset. Therefore, peers with similar offset share overlapping content focus, and data blocks can be exchanged among them. In order to locate random peers without centralize server, a distributed two-scale list which consists of S-peers and R-peers is proposed. With the help of R-peers, Peers can locate the desired peers when they perform VCR-like operations such as random seek. Considering urgency, rarity of chunks and asynchronous buffer, we propose a sophisticated demand-driven chunk swarming protocol to collaborate with ragged partners(S-peers). This protocol alleviates the workload on source server effectively.

The remainder of the paper is organized as follows. We present the background and review related work in section II. The playback time offset model is described in Section III. We discuss the demand-driven swarm model for P2P VoD system in section IV and present practical consideration for algorithm

implementation in section V. Then, we evaluate our scheme through simulation in section VI and conclude our findings in section VII.

II. Background and Related Works

Media service can be classified into two kinds: live and on-demand. Numerous application-level protocols have been developed for live streaming, which can be broadly classified into two categories according to their overlay structure, namely, *tree-based* and *mesh-based*. Similar to IP multicast, *tree-based* [2] [6] methods push data among users using a tree (or forest) rooted at the source. However, it is costly to maintain a *tree* structure; the streaming rate and robustness to network churn cannot be easily guaranteed.

Recently, inspired by successful file swarming system such as BitTorrent, data-driven mesh overlay is proposed to deliver live content to a large number of users, where coolstreaming [3] is a very important prior work. In data-driven approach, distribution path of chunks are dynamically determined based on chunk availability which is periodically exchanged. Therefore, this method can function effectively in dynamic environments (*e.g.*, peers may join/leave the system frequently, peer connections are with highly time-varying bandwidth). Alternative mesh structure [7] [8] and complicate segment schedule algorithm [9] have been presented to improve streaming quality.

Because of *shared temporal content focus*, peers in live streaming can typically exchange pieces effectively using a relatively small window. In contrast, peers may be at different playback points in the VoD system. This inbeing makes P2P VoD service more difficult. In tree-based P2P VoD system [10] [11] [12], Cache-and-relay applies the interval caching idea to address the asynchronous issue. With these techniques, each peer receives content from one or more parents and stores it in a local cache, from which it can later be forwarded to clients that are at an earlier play point of the file.

In the approach proposed by Annapureddy [13] for near on-demand streaming, each file is split into sub-files, which are encoded using distributed network coding and downloaded in a BitTorrent-like manner. Rather than splitting each file into sequentially retrieved sub-files, Toast [14] improve the VoD server by a video-targeted version of BitTorrent. P2P network in Toast act as distributed cache for VoD server. Probabilistic piece selection is adopted to support real-time delivery. Dana [15] propose a similar system called BASS and Pawel [16] use entropy as measure of bartering efficiency to analysis data-driven VoD system. Other proactive caching strategies [17] have been proposed to increase the number of replicas of each block, where blocks are altruistically replicated by peers not to aid immediate playback. PONDER [18] employs P2P downloading to serves a significant portion of data thereby reducing the load on the server, meanwhile, the server devotes its resource to offer viewers instantaneous playback and urgent data to meet performance requirement.

Since contents in clients' buffer are continuously changing and demands of different clients are asynchronous, finding partners with expected data and exchanging data with them is a very challenge problem. Dynamic Skip List(DSL) [19], Ring-assisted overlay(RINDY) [20] are proposed for VCR interaction, Chi [21] presents a buffer-assisted search(BAS) scheme to improve search efficiency by reducing the size of index overlay. A similar idea has been adopted in this paper, where playback offset is an important guidance to locate partners.

III. Playback time offset ranking

As peers join VoD system at any given time and watch the video from any points, an effective playback offset model needs to be used as service discovery mechanism. In this section, we propose our playback offset model and provide an effective partner discovery method for demand-driven chunk swarming which is described in section IV.

Prior to give detail illustrator of this model, several terminologies are introduced here. Length of the stored media is L, and it is partitioned into M equal-size chunks. Users are arranged by their arrival order, *i.e.* node n_i is the *i*-th user joined the system. n_i join the system at t_i and the start point of n_i is v_i. For example, v_i is set to 0 for most users who watch the video from the beginning. $P_i(t)$ is the playback offset of user i at time t; T_m is the playback time of the m-th data segment. Therefore, the playback duration of the m-th data segment is $T_{m+1} - T_m$, and the last data segment is $L - T_M$.

In a P2P VoD system, peers cache received chunks around their playback offset in their buffer; earlier users can upload data in their buffer to later users. Given a limited buffer size B, peers need to locate collaborators with overlapping buffer in order to barter chunks with them. Without VCR interaction, the rank value of $P_i(t)$ is stable over time since play rate is identical for all normal client. Initially, peers can fetch collaborators from the rendezvous point when they join the system. Users may (1) stop watching the video for a moment and resume after they finish another task; (2) seek randomly to preview the video and watch the most interested part according to personal taste. Pause, resume and random seek are three common VCR interactions due to users' behavior. Consequently, it is difficult to rank $P_i(t)$ in time for sake of random VCR interaction.

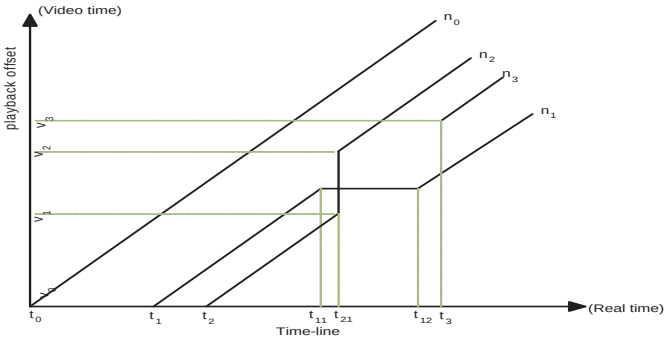


Fig. 1. Users arrive at any time and watch video from any position

Figure (1) show an example, n_0, n_1, n_2, n_3 join the VoD

system at t_0, t_1, t_2, t_3. n_0 play the video in normal way without VCR operation. The playback offset of node n_i is 0 when n_i has not joined the system. As $t < t_{11}$, the rank value of $P_i(t)$ is $P_0(t) \geq P_1(t) \geq P_2(t) \geq P_3(t)$. Then node n_1 pause at t_{11} for $t_{12} - t_{11}$ time unit and resume at time t_{12}, n_2 jump from v_1 to v_2 and n_3 watch the video from v_3. The rank value changes to $P_0(t) \geq P_2(t) \geq P_3(t) \geq P_1(t)$. The collaboration is destroyed by VCR interaction. In P2P live streaming systems, network failure and peers' departure can break users' collaboration, and failure recovery component is an essential part of practical live streaming systems. Similarly, network failure, peer churn, user interaction and asynchronous requests need to be addressed in a P2P VoD system.

In this paper, we introduce a distributed two-scale list to manage playback offset. Each peer in the system maintains some peers(S-peers) with similar playback offset and some remote peers(R-peers). S-peers are used for chunk swarming and R-peers are used to support VCR interaction. Figure 2 is an example, where s_1, s_2, s_3, s_4 are n_i's S-peers and v_1, v_2 are n_i's R-peers.

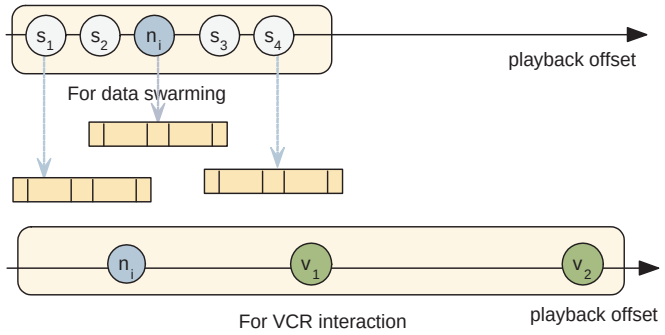


Fig. 2. Two-scale list of members according to playback offset

Suppose that node n has R multi-resolution R-peers, where the playback offset distance between n and R_i is d_i, ideally, $d_i = ca^i$, where c is constant and a is the multiplicative increase factor. It is impossible to find these perfect R-peers when peer join the VoD system randomly. We choose a peer whose d_i is closest to the anchor point as R_i. If n needs to find peers whose playback offset is close to T, it send lookup message to R_k. Without loss of generality, we suppose that node n jump forward, i.e. $T > P_n(t)$. k is calculated by equation (1).

$$d_k \leq T - P_n(t) < d_{k+1} \tag{1}$$

Then, R_k checks whether T locates in S-peers' BW. If yes, R_k send these S-peers back to n. Otherwise, the lookup procedure is performed by k. The lookup procedure is terminated when R_k find desired S-peers or no R-peers satisfy equation 1. In the later case, There are no suitable collaborators for n in R_k's view, and it selects its S-peers randomly to send them back.

IV. DEMAND-DRIVEN SWARM MODEL

Depend on playback offset model, peers can form their collaborating group. Consequently, peers should collaborate with their partners and schedule data transmission according to their buffer status and bandwidth utility. The peer/chunk selection protocol consists of two parts: (1) determine the sequence of data blocks to be retrieved, (2) determine from which partner to get the data. In this section, we will firstly introduce some basic data schedule protocols and then present our demand-driven swarm data schedule in consideration of VoD service.

To make our discussion easier, several realistic assumptions are given here. To maximize the utilization of incoming and outgoing access link bandwidth, *bandwidth-degree condition* [7], i.e. the same bandwidth to degree ratio, is satisfied in our VoD overlay. Peer n_i allows U_i concurrent upload connections and D_i concurrent download connections. Buffer map [3], which is called BM for abbreviation in this paper, is exchanged among partners periodically. n_i can only request for missing segments within the buffer window according to the notification of data availability, and sends the available data to its neighbors who ask for it. To make the streaming more continuous, n_i should tolerate some delay at the beginning as it fulfills the buffer before starting to playback.

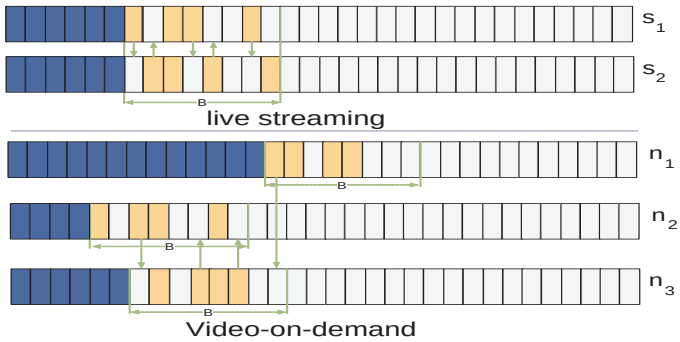


Fig. 3. Chunk schedule in live streaming and VOD

Generally speaking, there are four kinds of chunk selection protocol proposed:

- **Pure random strategy :** Peers select the missing chunks randomly to request them from their neighbors.
- **Strict in-order strategy :** Peers aim to fill the empty buffer location closest to the playback time firstly for timely delivery. This strategy seems intuitively for media service in consideration of deadline requirement. Zhou [22] points out that this strategy is too short-sighted from system view.
- **Local rarest-first strategy :** Peers fetch the rarest chunks firstly. This strategy maximum the chunk diversity and produce good throughout, and it is widely used in Bit-Torrent.
- **Mixed strategy :** The first part of chunks are scheduled by in-order strategy and the other part is scheduled by rarest-first strategy. This combined strategy makes a tradeoff between playback continuity and swarming efficiency.

Figure 3 depicts the difference of chunk swarming between

live streaming and VoD system, where s_1, s_2 are nodes in live streaming system and n_1, n_2, n_3 are nodes in VoD system. s_1, s_2 focus on the some video region and a broadcast-like chunk dissemination is enough from system view. Some simple chunk schedules have been deployed in commercial live system and seemingly are able to provide good QoE [23]. Considering BM of n_1, n_2, n_3, chunk schedule is much complicate in VoD system. We consider three factor (urgency, rarity of chunks and asynchronous BM) in order to maximize the number of segment that can be retrieved from the partners before the playback (urgent) deadline.

Periodically, nodes exchange buffer status with their S-peers, and they get the available chunk information. Let $B(n)$ denote the BM of node n and $S(n)$ denote the set of chunks which node n has received. Chunks that can be retrieved by node n from its S-peers are calculated by equation (2).

$$W(n) = \bigcup_{i \in S-peers\ of\ n} S(i) \backslash S(n) \tag{2}$$

Urgency of chunk m in this schedule period t is calculated by equation (3), where $id(m)$ is the sequence number of chunk m in BM, k is the chunk being used, i.e. $P_n(t) \in [T_k, T_{k+1})$.

$$U(m) = id(m) - id(k) \tag{3}$$

Rarity of chunk m is calculated by equation (4).

$$\begin{aligned} R(m) &= \sum_{i \in S-peers\ of\ n} f(m, S(i)), \\ f(m, S) &= \begin{cases} 1, & m \in S, \\ 0 & m \notin S \end{cases} \end{aligned} \tag{4}$$

In figure 3, there is an overlap of one chunk between node $n1, n3$. The only swarm opportunity between $n1, n3$ is that $n3$ pulls the last chunk in its buffer window from $n1$. In the high dynamic VoD system (VCR interaction or peer churn), we propose a greedy strategy to grant high priority to pull chunks from small opportunistic S-peers. Asynchronous gain $A_k(m)$, which is calculated by equation (5), is used to determine the system utility when node n retrieve chunk m from S-peer k.

$$A_k(m) = \begin{cases} \|B(n) \cap B(k)\|, & m \in S(k) \backslash S(n), \\ 0 & others \end{cases} \tag{5}$$

And,

$$A(m) = \max_{k \in S-peers\ of\ n} \{A_k(m)\} \tag{6}$$

We combine the three terms above to give priority of the missing chunks in each schedule period. i.e.

$$\begin{aligned} Priority(m) =& \alpha U(m) + \beta R(m) + \gamma A(m) \\ & \alpha + \beta + \gamma = 1 \end{aligned} \tag{7}$$

And we send request to get low priority value chunk m from low $A(m)$ S-peers firstly. If peers cache all the chunks they have received, they can reduce server load greatly in two modes. (1) As peers randomly seek backward, they can fetch data from local disk of themselves without any network traffic. (2)Peers can redistribute chunks they cached to later users (called single video approach in [24]). In our approach, peers fetch data from earlier users or from the source server directly if demand-driven chunk swarming cannot satisfy their playback deadline, which guarantees QoE of end users.

V. Practical Consideration

In this section, we discuss how to incorporate playback offset rand with demand-driven swarming to support playback quality and VCR operation. A general system model is illustrated in Figure (4).

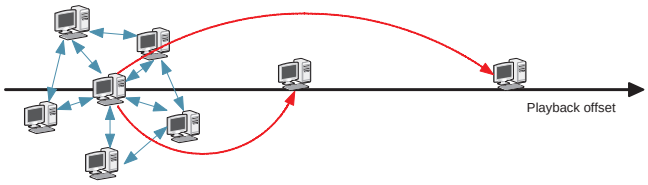


Fig. 4. Simple description of our VoD system model

A. peer churn

The source server records the join time of all the peers in the system, which is called *tracker* in BitTorrent. When a new peer joins the system, it submits its start point to the tracker. Tracker selects S-peers and R-peers for the new member according to its initial playback offset. Noting that S-peers may be empty due to sparse join time, if the new peer receives empty S-peers, it request chunks directly from the source server or from earlier peers. Otherwise, the peer collaborates with its S-peers to fetch data in demand-driven model described section IV, and it updates its two-scale list by exchanging with its S-peers and R-peers.

In a dynamic P2P network, peers may depart unexpected at any time gracefully or ungracefully due to failure. If peer A leaves VoD system gracefully, it sends its leaving message to its S-peers and the tracker. *Heart beat* message is used to detect ungracefully departure. If A doesn't receive *heart beat* from B for a long time, A considers that B has left the system and removes B from its partner list. A can absorb the impact caused by $B's$ departure if B is a S-peer because A updates its S-peers periodically. If B is a $R-peer$, A needs to find a substitution just like performing a random seek operation described next.

B. VCR interaction

We consider three common VCR operations in this paper, i.e. pause, resume and random seek. When users perform VCR operation, their playback offset is affected and two-scale list needs adjusting for chunk swarming and future VCR interactions.

1) pause and resume: Users perform pause and resume operation in two scenarios: (1) There is another task that needs finishing firstly, (2) They can tolerate more playback delay than playback continuity and buffer more chunks in memory due to poor network condition. It is very simple to support pause/resume operation in our method. As users pause for a moment, chunks are still swarmed while playback offset is not

updated. In order to keep up with our demand-driven chunk swarming model, we introduce **V**irtual **P**layback **O**ffset(*vpo*) and **A**ctual **P**layback **O**ffset(*apo*) here. *vpo* is the start point of BW and *apo* is the playback offset of the media player. In system view, the playback offset of a peer is $\max\{apo, vpo\}$. Peers drain chunks in their buffer when they resume watching the video.

2) random seek: Users perform seek backward/forward interaction randomly when they want to watch special part of the video. Suppose that peer n jumps forward δ at time t, Therefore, its playback offset changes to $P_i(t) + \delta$. If δ is small enough and $P_i(t) + \delta$ locates in some S-peers' BW, n triggers update of its S-peers by requesting partner information from these S-peers. Otherwise, n triggers a lookup procedure to find some new S-peers close to the target position.

VI. Performance Evaluation

We use simulations to evaluate the performance of our proposed algorithm and the factors that affect it. For the sake of comparison, we select three representative schemes which corporate P2P spirit into VoD service. The first is Toast [14] which improves the VoD system by a video-targeted BitTorrent. The second is Ponder [18] which combine P2P downloading with traditional client-server VoD system. The third is called sub-file in comparison, which is proposed by Annapureddy [13]. We don't consider network coding for comparison because it is out the scope of our discussion.

In the following evaluation and comparison, we consider three metrics:

- *source server load:* which is the number of the chunks transferred by the source server and is the major metric in P2P VoD system.
- *playback continuity:* which is the ratio of chunks that arrived at peers ahead of their urgent deadline.
- *VCR overhead:* which is the number of peers that has been contacted to locate desired partner.

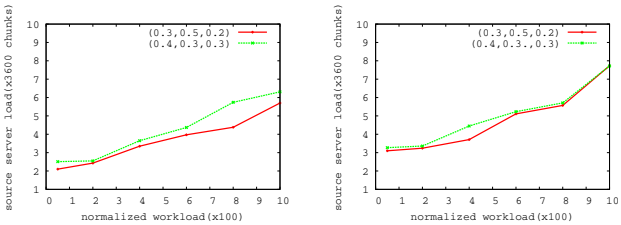
A. Experiment setup

In the simulation experiment, we consider two scenarios: flash crowd scenario and Poisson arrival scenario. The media length L is set to 3600 seconds and each chunk is of 1-second (in sub-file scheme, we divide media file into 30 sub-files), which is the minimum data unit. The BM is set to 60 chunks, i.e. every peer can buffer one minute video for swarming. The population of users in VoD system ranges from 50 to 1000. The total number of clients joining the system during the simulation is called normalized workload [11]. In flash crowd scenario, all these users join the system in the first five minutes uniformly. With Poisson arrival pattern, the average inter-arrival time is set to 3 second. The source server can send 10 chunks per second. There are two kinds of peers with different combination of download bandwidth and upload bandwidth. The first type is with 1 chunk/second download rate and 0.5 chunk/second upload rate, the second type is with 2 chunks/second download rate and 2 chunks/second upload rate, and 70% of VoD users are the second type.

B. Simulation Results

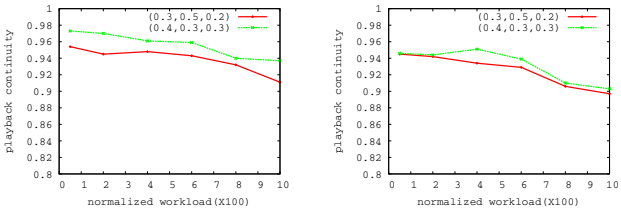
We firstly simulate two different configurations to examine how the weights (α, β, γ) in demand-driven chunk swarming model affects the performance. Configuration 1 is $\alpha = 0.3, \beta = 0.5, \gamma = 0.2$ and configuration 2 is $\alpha = 0.4, \beta = 0.3, \gamma = 0.3$.

Figure 5 shows the results of the source server load. The server only needs to send the stored video 5.3 times as 1000 users join in flash crowd scenario and 7.2 times in Poisson arrival scenario. Configuration 1 performs better in reduction of server load than configuration 2 in flash crowd scenario, and the performance gap between these two configurations is diminished in Poisson arrival pattern. Therefore, sparse arrival pattern can reduce the influence of weights (α, β, γ) in chunk swarming schedule. Figure 6 considers the playback continuity, approximately, 96.5% chunks of configuration 1 and 93.4% of configuration 2 arrive in time in flash crowd while 92.4% chunks of configuration 1 and 91.8% chunks of configuration 2 arrive in time in Poisson arrival. Average VCR overhead of random seek is depicted in Figure 7 where K is the number of R-peers, results show that peers can locate suitable collaborators quickly.



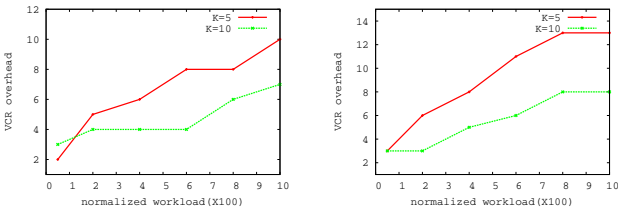
(a) Flash crowd (b) Poisson arrival

Fig. 5. Server load under different α, β, γ



(a) Flash crowd (b) Poisson arrival

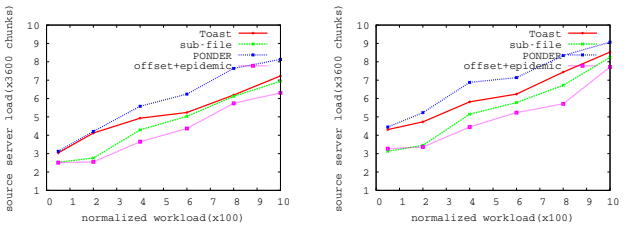
Fig. 6. Playback continuity under different α, β, γ



(a) Flash crowd (b) Poisson arrival

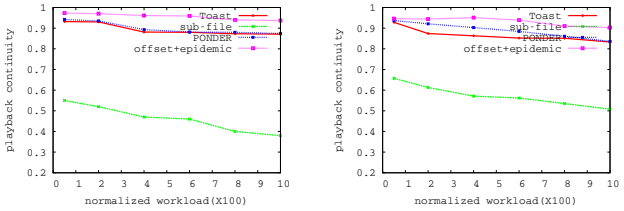
Fig. 7. Overhead of VCR interaction

Figure 8 and Figure 9 show the results of comparison of our method and other three scheme under study. Among the four schemes, our method outperforms other three in terms of server load and playback continuity. The playback continuity of subfile without network coding is under 0.5 and it cannot provide good QoE for a practical VoD system. Accordingly, urgency aware chunk pull method is critical for sake of continuity in P2P VoD system.



(a) Flash crowd (b) Poisson arrival

Fig. 8. Server load under different schemes



(a) Flash crowd (b) Poisson arrival

Fig. 9. Playback continuity under different schemes

VII. Conclusion

We propose two-scale list and demand-driven chunk swarming to provide VoD service with high quality of experience to end users. With R-peers defined in this paper, peer performing VCR operation can locate desired partners with low overhead. To maximize the swarming opportunity, we proposed a demand-driven chunk swarming model. Our simulation results show that the source server load is significantly transferred to peers who are watching different portion of the video. Because of its efficiency and easy deployment, our method is practical for P2P VoD system.

References

[1] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, (New York, NY, USA), pp. 15–28, ACM, 2007.

[2] C. Yang-hua, S. G. Rao, S. Seshan, and A. H. Z. Hui Zhang, "A case for end system multicast," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1456–1471, 2002.

[3] Z. Xinyan, L. Jiangchuan, L. Bo, and Y. S. P. A. Y. Y. S. P. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *INFOCOM 2005*, vol. 3, pp. 2102–2111 vol. 3, 2005.

[4] PPLive, "http://www.pplive.com."

[5] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson, "Analysis of bittorrent-like protocols for on-demand stored media streaming," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 1, pp. 301–312, 2008.

[6] C. Miguel, D. Peter, K. Anne-Marie, N. Animesh, R. Antony, and S. Atul, "Splitstream: high-bandwidth multicast in cooperative environments," 2003. 945474 298-313.

[7] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," in *INFOCOM 2007*, pp. 1415–1423, 2007.

[8] F. Wang, Y. Xiong, and J. Liu, "mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *Distributed Computing Systems, 2007. ICDCS '07. 27th International Conference on* (Y. Xiong, ed.), pp. 49–49, 2007.

[9] M. Zhang, Y. Xiong, Q. Zhang, and S. A. Y. S. Yang, "On the optimal scheduling for media streaming in data-driven overlay networks," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE* (Y. Xiong, ed.), pp. 1–5, 2006.

[10] G. Yang, S. Kyoungwon, K. Jim, and T. Don, "P2cast: peer-to-peer patching for video on demand service," *Multimedia Tools Appl.*, vol. 33, no. 2, pp. 109–129, 2007. 1236532.

[11] T. T. Do, K. A. Hua, and M. A. Tantaoui, "P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Communications, 2004 IEEE International Conference on* (K. A. Hua, ed.), vol. 3, pp. 1467–1472 Vol.3, 2004.

[12] C. Yi, L. Baochun, and K. Nahrstedt, "ostream: asynchronous streaming multicast in application-layer overlay networks," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 1, pp. 91–106, 2004.

[13] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez, "Is High-Quality VoD Feasible using P2P Swarming?," in *Proceedings of the 16th International World Wide Web Conference (WWW2007)*, (Banff, Canada), May 2007.

[14] Y. R. Choe, D. L. Schuff, J. M. Dyaberi, and V. S. Pai, "Improving vod server efficiency with bittorrent," in *ACM Multimedia*, pp. 117–126, 2007.

[15] D. Chris, L. Danjue, H. David, and A. C.-N. C. Chen-Nee Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *Multimedia Signal Processing, 2005 IEEE 7th Workshop on* (L. Danjue, ed.), pp. 1–4, 2005.

[16] P. Garbacki, D. H. J. Epema, J. Pouwelse, and M. van Steen, "Offloading servers with collaborative video on demand," in *7th International Workshop on Peer-to-Peer Systems (IPTPS'08)*, (Tampa Bay, FL), February 2008.

[17] N. Vratonjić, P. Gupta, N. Knežević, D. Kostić, and A. Rowstron, "Enabling dvd-like features in p2p video-on-demand systems," in *P2P-TV '07: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, (New York, NY, USA), pp. 329–334, ACM, 2007.

[18] G. Yang, Y. Shengchao, L. Hang, S. A. M. S. Mathur, and K. A. R. K. Ramaswamy, "Supporting vcr operation in a mesh-based p2p vod system," in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE* (Y. Shengchao, ed.), pp. 452–457, 2008.

[19] D. Wang and J. Liu, "A dynamic skip list-based overlay for on-demand media streaming with vcr interactions," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 4, pp. 503–514, 2008.

[20] B. Cheng, H. Jin, and X. Liao, "Supporting vcr functions in p2p vod services using ring-assisted overlays," in *Communications, 2007. ICC '07. IEEE International Conference on* (H. Jin, ed.), pp. 1698–1703, 2007.

[21] C. Huicheng, Z. Qian, J. Juncheng, and A. X. S. Xuemin Shen, "Efficient search and scheduling in p2p-based media-on-demand streaming service," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 1, pp. 119–130, 2007.

[22] Z. Yipeng, C. Dah Ming, and J. C. S. Lui, "A simple model for analyzing p2p streaming protocols," in *ICNP 2007* (C. Dah Ming, ed.), pp. 226–235, 2007.

[23] C. Liang, Y. Guo, and Y. Liu, "Is random scheduling sufficient in p2p video streaming?," in *ICDCS 2008*, vol. 0, (Los Alamitos, CA, USA), pp. 53–60, 2008.

[24] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable?," in *SIGCOMM 2007*, (New York, NY, USA), pp. 133–144, ACM, 2007.