Differentiated Chunk Scheduling for P2P Video-on-Demand System

Ubaid Abbasi[1], Gaétan Simo[2] and Toufik Ahmed[1]
[1]CNRS LaBRI Lab. – University of Bordeaux, France {abbasi, tad}@labri.fr
[2]ENSEIRB MATMECA Bordeaux, France {simo@enseirb-matmeca.fr}

Abstract*—The peer-to-peer (P2P) networks provides a data distribution model that is attractive for Video on Demand (VoD) as it allows to decrease the costs and increase the scalability of video distribution. The two significant challenges in P2P VoD streaming are scalability and video quality. Both require efficient utilization of the resources in P2P network. Inspired by this finding, the paper addresses the problems of chunk scheduling and bandwidth allocation in P2P VoD system to efficiently utilize the upload bandwidth capacity of the peers. We first propose a queue based chunk scheduling mechanism followed by a bandwidth distribution algorithm for urgent downloading and prefetching. Our proposed mechanism allows the maximum utilization of upload bandwidth by distributing the available upload bandwidth among different queues. Experimental results show that differentiated queuing is capable of achieving the optimal streaming rate.

Keywords: P2P Network, chunk scheduling, Bandwidth allocation, Overlay Organization, Quality of Service (QoS).

I. INTRODUCTION

Streaming applications have recently attracted a large number of users on the Internet. In 2008, the number of video streams served increased 24.3% to 41.6 billion even without counting the user generated videos [1]. With the fast deployment of high-speed residential access, video traffic is expected to dominate the Internet in near future. To meet the demand of explosively growing multimedia applications, media streaming has been a research topic attracting significant interests over the past two decades. The ultimate goal of Internet media streaming is to satisfy the application requirements of as many end users as possible, with sustainable server bandwidth costs. The traditional client/server architecture advocates the use of large data centres to maintain streaming to end users at a large scale. The bandwidth cost on servers increases rapidly as the user population increases, and may not be manageable in corporation with limited resources.

IP multicast [2][3] and content delivery networks (CDNs), attempted to tackle the problem by conserving resources in the edge or core routers, or by load balancing across a large number of edge servers. However, the problem of scalability to a large user population in media streaming systems is only mitigated to a certain degree, not solved.

Over the last few years, Peer-to-Peer (P2P) networks have emerged as a promising approach for distribution of multimedia contents over a large scale network [4]. P2P networks propose a different architectural design perspective. It offloads part of the bandwidth burden from dedicated streaming servers hosted by the content providers, and shifts them to end hosts themselves when they serve content to each other. The P2P design philosophy seeks to utilize peer's upload bandwidth for reducing server's workload. However, the upload bandwidth utilization might be suppressed by the so called *content bottleneck* where a peer may not have any content that can be uploaded to its neighbors even if its link is idle. The content bottleneck causes more severe problems in VoD system, due to free user's control (forward, resume, pause etc). To make things worst peers are interested only in a small portion of chunks and their priorities changes more frequently as compared to live streaming. One way to resolve this problem is to compromise user viewing quality. For example, a lower video playback rate has lower peer bandwidth utilization requirement. Allowing a longer playback delay also allows a larger set of chunks to be exchanged among peers. The other solution lies in designing more efficient prefetching strategies and chunk scheduling methods.

In this paper, we propose a differentiated chunk scheduling mechanism that can achieve high peer bandwidth utilization. Using queue-based signaling between peers and the content source server, the amount of workload assigned to a peer is proportional to its available upload capacity, which leads to high bandwidth utilization. In VoD system, the chunks closer to the current playing position have more importance, therefore a queuing model is designed for the segregation of "urgent" and "prefetching" traffic in VoD system. More specifically our paper provides following three fold contributions.

I. We investigate the server's side of peer, and classified the content requests into separate queues. We then proposed different scheduling policies for these queues considering the importance of each type of chunk.
II. We proposed a link sharing mechanism, to prioritize the "urgent downloading" target. The bandwidth sharing among the queues therefore follows a logical pattern.
III. We evaluate the properties of our algorithms, through real test bed.

II. Related Works

Several recent works proposed a centralized solution that can fully utilize peer's uploading bandwidth and achieve the streaming rate upper bound [5]. The centralized solution collects all peers upload capacity information, and calculates the sub-stream rates sent from the server to peers. In practice, available upload capacity varies over time and peers join and leave the system. The central coordinator needs to continuously monitor peer's upload capacity and re-compute the sub-stream rate to individuals. This results in excessive computation and overhead on a single server.

The earliest work looking at improving download time is by Bernstein *et al.* [6] on adaptive server peer selection based on server attributes and partial downloads. The authors propose, using machine learning techniques for clients, to adaptively select among alternative servers in order to reduce the download time. While smarter server selection at the client side may result in faster downloads, many of the available peers will probably be highly popular ones that are overloaded, due to both low (constrained) resource availability and a large number of queued download requests.

There have been ongoing efforts intending to improve resource utilization in P2P streaming systems. The study in [7] shows the mesh-based scheme can better utilize peer's upload capacity than tree-based scheme, due to the dynamic mapping of content to the delivery paths. To improve the resource utilization in mesh-based P2P streaming, authors in [8] propose a multi-phase swarming scheme where the fresh content is quickly injected to the entire system in the first phase, and peers exchange available content in the second phase.

Network coding is also applied to P2P live streaming. In [9], authors perform a reality check by using network coding for P2P live streaming however, neither approach can fully utilize the resources and achieve the maximum streaming rate. The authors in [10] give a randomized distributed algorithm that can converge to the maximum streaming rate. They also study the delay that users must suffer in order to play the stream with a small amount of missing data.

In [11] authors describe architectural design issues of a real P2P VoD system. The author also points out the departure misses which are the major cause of server load. In another similar work [12], the author proposes an aggressive replication policy to reduce departure misses. A peer can proactively replicate popular chunks to other peers no matter whether they need these chunks or not. While this replication policy may reduce server load, it also severely wastes precious peer upload bandwidth.

In our previous work [14], we proposed a cooperative prefetching technique. In this strategy, the requested segments in VCR interactivities are prefetched into session beforehand using the information collected through gossips. The peers in the same session exchange the information related to available segments. The segments which are not available in the session are fetched from other sessions. This technique reduces the delay and improves the hit ratio.

There are several fundamental questions that are unclear such as how to differentiate between different request types, which chunks should be given priority and what the limitations of scheduling are and its trade-offs. The existing works didn't provide a comprehensive study on these crucial issues. In differentiated chunk scheduling, we focus on maximizing the utilization of upload bandwidth of peers. We attempt to provide an effective scheduling mechanism for P2P VoD system that assigns different priorities to different request types. On the basis of these priorities video chunks are provided to peers.

The remaining part of this paper is organized as follows. The queuing model and scheduling algorithm of distributed chunk scheduling are described in Section III. The experiment results are reported in Section IV. Finally, section V ends the paper with concluding remarks and an insight on future work.

III. Distributed Chunk Scheduling

The ability to achieve higher streaming rate in P2P VoD system is highly desirable. Higher streaming rate provides better quality and perception of stream. It also provides a cushion to absorb the bandwidth variations caused by peer churn and network congestion. The key to achieve high streaming rate is to better utilize the peer's upload bandwidth.

In this section we propose a differentiated chunk scheduling mechanism that can achieve maximum upload bandwidth of peers in P2P networks. We discuss the scheduling mechanism when peer is acting as a content source or content provider (server side scheduling). We assume a fully connected mesh topology, in which peers sends pull request to obtain the desired content from other peers or server. The availability of upload capacity is conditional to the queue status.

The following sub-sections will explain in detail the proposed differentiated chunk scheduling policy.

A. Server Side Scheduling

The queuing model is specifically designed for the case of co-existence of "*urgent downloading*" and "*prefetching*" requests on each peer. *Prefetching has been proposed as a technique for reducing the access latency*. In this technique, peers prefetch and store various portions of the streaming media ahead of their playing position.

Each peer in the overlay providing the content to other peers is considered as content server in our case. We used two different queues for two different types of requests. Before sending a request for chunk, each peer sets an identifier for making distinction between the two types of content requests. On each peer (content server), there is a classifier which checks the *request-type* and sends it to appropriate queue. The urgent downloading target requires higher priority because the requested chunks are closer to the current position of playing window. There is also a scheduler which determines the order of packets to be transmitted from the queues. Figure 1 shows the model of queue based chunk scheduling. In this figure, the serving peer receives different types of chunk requests. These requests are classified into different queues according to their identifier.

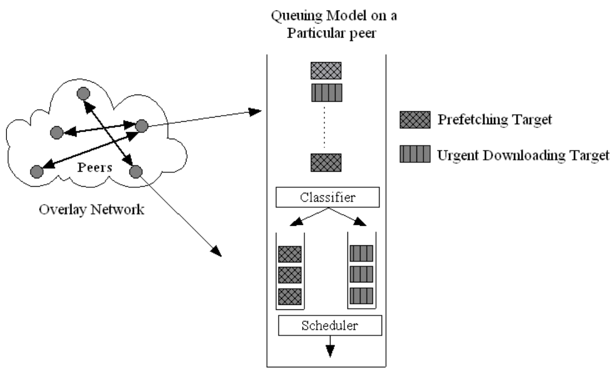

Figure 1: Queuing model in a particular Peer

We used two different types of scheduling policies for each queue. For urgent downloading target, the chunks whose deadline is near to expire have given priority. Thus *earliest deadline first* (EDF) is adapted in this case. This allows the timely availability of chunks to the requesting peer. If a peer requests multiple chunks at different time interval, the latest request will be served while the earlier chunk request would be forwarded to prefetching queue. Let two chunks are requested from a same peer at time t_e and t_c, where t_e is earlier time and t_c denotes the current time. If the difference between the two time (t_e and t_c) is greater than certain threshold, then this situation suggests that peer has performed a seek operation and now it's playing position have been changed. We define the time threshold equals to 10 seconds which is same as the length of window for urgent downloading [13]. In this case, we give priority to the chunk closer to current position, thus the latest request (at time t_c) has been fulfilled. This scheduling scheme allows the peer to obtain the chunks nearer to playback position.

On the other hand we used simple *first come first serve* (FCFS) policy for prefetching queue. This type of content doesn't have a specific deadline. These content are used to reduce the delay latency when a user performs a seek operation. Therefore FCFS policy is sufficient for this type of content.

B. Bandwidth Distribution

We design a scheduler to determine the order of packets to be transmitted from the queues according to the bandwidth ratio "*br*" for each type of traffic. *The bandwidth ratio "b_r" represents the amount of bandwidth dedicated to urgent downloading and prefetching.*

```
Input:
        $G = (V, E)$
        Chunk request: $r_{i,j}$ for $i, j \in V$
        Time Interval : $t_i$
        Upload Bandwidth: $\mu_i$ for $i \in V$
        Video Chunk : $c$
Output:
        Video Chunk Schedule;
Algorithm:
1.   for each $r_{i,j} \in R$
2.     if (ReqType = Urgent)      //Urgent download
3.       Push $r_{i,j}$ to UrgentQueue
4.       Update UrgentQueue ($r_{i,j}$)
5.     else
6.       Push $r_{i,j}$ to PrefetchQueue
7.   for each $t_i \in T$
8.       Calculate bandwidth ratio
9.   for each $r_{i,j} \in$ UrgentQueue
10.      Sort $r_{i,j}$ according to deadline
11.      Push $C_{j,i}$ to i
12.  If ($\mu_i$- br > 0)
13.      for each $r_{i,j} \in$ PrefetchQueue
14.          Sort $r_{i,j}$ according to FCFS
15.          Push $C_{j,i}$ to i
```

Figure 2: Algorithm for Queue and Bandwidth Distribution

Moreover, both classes can borrow bandwidth from each other when one of the two types of traffic is non-existent or under the limit. This *br* value is also used to calculate the service rate for both types of traffic on that particular peer with br_i and μ_i- br_i being respectively the service rate for urgent downloading and prefetching for peer *i*. μ_i is the total available bandwidth of peer *i*. In order to calculate the value of *br* we monitor the first queue (urgent downloading) in periodic interval. We calculate the total size of data chunks requested and their corresponding deadlines. Let CS_i represents the chunk size requested by peer *i* with deadline t_i then,

$$\text{Bandwidth ratio } (br_i) = \frac{\sum_{i=0}^{n} CS_i}{\sum_{i=0}^{n} t_i}$$

This value of *br* is used to distribute the upload capacity of the peer among the two types of traffic. The urgent downloading target has higher priority therefore br_i is the outgoing capacity of this link. The remaining bandwidth μ_i- br_i is assigned to the prefetching queue. The peers upload bandwidth doesn't remain constant and fluctuates over time. The periodic calculation of the bandwidth ratio allows to handle the dynamicity of the network. The algorithm for differentiated chunk scheduling is described in Figure 2.

C. *Client Side Scheduling*

We divide the client buffer window into two different stages, according to play back time of segments as shown in Figure 3. The client side structure is similar to most P2P VoD implementations [13]. The *adjacent stage* contains the segments which are more closer to the current playing position of the window. Thus the segments in this window are considered extremely important and therfore given higher priority. The prefetching stage contains the block with the latest playback time. We utilize cooperative prefetching[14] to prefetch the content from different peers. This technique fetches the maximum unavailable segments into session thus reducing the inter-session transfer delay. The other segments to be prefetched are given lower priority as a request identifier.

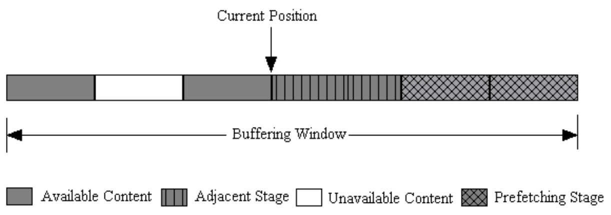

Figure 3: Sliding Window in VoD System

IV. Performance Evaluation

This section describes the performance evaluation of differentiated queuing mechanism for different QoS parameters using real test bed.

A. *Experimental Setup*

We examined the performance of differentiated queue scheduling through experiments on a real network. We implemented the prototype of our proposed mechanism on a typical VoD system, with a tracker, content source and different peers. The tracker provides the initial list of sources (seeders) to a new arriving peer. The new peer exchanges the necessary control information to start receiving the content. All connections between the peers are TCP connection. For bandwidth distribution we used the data suggested in [15]. Internet has the characteristic of rich diversity [16][17] and that's true in our case for end nodes (peers). All the peers are operating in surplus mode having enough bandwidth for urgent downloading and prefetching.

We compare the performance of differentiated queue scheduling with a single queue mechanism. The single queue mechanism utilizes same queue for two types of requests. Moreover the prefetching requests from the peers are also randomly generated.

Performance Metrics: The performance evaluation is carried for different QoS metrics that include: bandwidth utilization, latency and video throughput. These parameters have significant role in determining the overall QoS for the VoD streaming applications.

B. *Results and Discussion*

The average latency for both mechanisms is given in Figure 4. We measure the average arrival time of packets for both mechanisms. The x-axis shows the position of playing window in Megabits while y-axis depicts the average latency. It is observed that differentiated queue scheduling with cooperative prefetching has greater delay initially. This is understandable because our mechanism focus on prefetching rare chunks into session and later on if any peer need a certain chunk, it can prefetch from a peer in same session, with small delay. The latency tends to decrease as the video progresses due to the presence of sufficient chunks for seek operations.

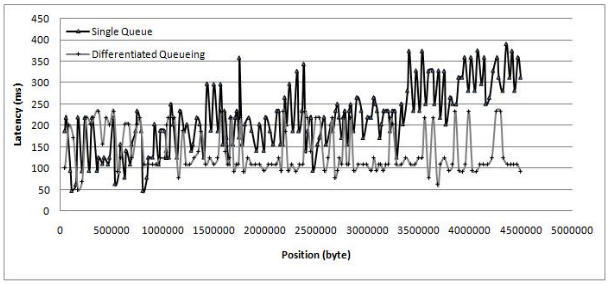

Figure 4: Comparison of Average Latency

Figure 5 shows the rate achieved by differentiated queuing compared to optimal rate. We calculate the optimal rate using formula given in [18]. The difference never increases 8% of the optimal rate possible in the system. The curve exhibits variation due to the various mode of operation. When the system is working in surplus mode (bandwidth required is less than bandwidth available) the achievable rate decreases. However when system is working in deficient mode, maximum upload bandwidth is utilized and optimal rate is achieved at some points.

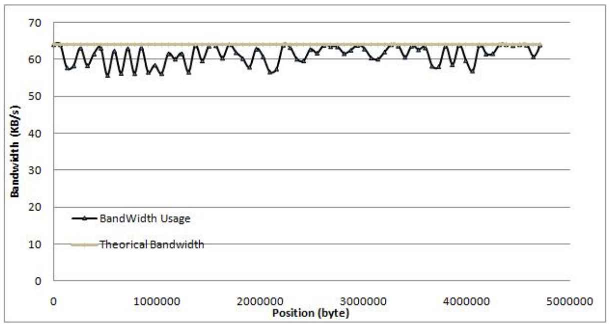

Figure 5: Achieved rate Vs Optimal rate

V. CONCLUSION & FUTUR PERSPECTIVES

In this paper, we propose a simple differentiated chunk scheduling mechanism that can achieve maximum bandwidth utilization in P2P VoD system. To study the effectiveness of the proposed mechanism, a prototype is developed and is used to conduct experiments over the real network. The results demonstrate the optimality and the effectiveness of the proposed chunk scheduling mechanism.

Future work can develop along several directions. As the first attempt of applying differentiated queue management to P2P VoD system, we used simple queue distribution schemes. We will explore queue control design space to further improve the performance of chunk scheduling mechanism. Secondly, we did not compare the performance of differentiated chunk scheduling with other existing methods. Although we are confident that the proposed scheduling mechanism can outperform existing approaches due to its optimality, simplicity, and flexibility, it will be an interesting exercise to do the comparison with existing solutions.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) in the ENVISION project, grant agreement 248565.

REFERENCES

[1] “Accustream iMedia Research Homepage,” 2009. http://www.accustreamresearch.com

[2] Xing Jin, Cheng Kan-Leung, Chan S.-H. Gary, “Island Multicast: Combining IP Multicast With Overlay Data Distribution”, “IEEE transactions on multimedia , 2009, vol. 11, no.5, pp. 1024-1036

[3] Xing Jin, Cheng Kan-Leung, Chan S.-H. Gary, “Scalable island multicast for peer-to-peer streaming”, Hindawi Publishing Corporation, Advances in Multimedia, Volume 2007, Issue 1, ISSN:1687-5680.

[4] Mubashar Mushtaq, Toufik Ahmed, and Djamal-Eddine Meddour “Adaptive Packet Video Streaming Over P2P Networks” in ACM International Conference Proceeding part of International Workshop on Peer to Peer Information Management (P2PIM), Vol. 152, Article N°59, Hong Kong, May 2006.

[5] R. Kumar, Y. Liu, and K. Ross, “Stochastic fluid theory for p2p streaming systems,” in *Proceedings of IEEE INFOCOM*, 2007.

[6] Bernstein, D. S., Feng, Z., Levine, B. N., Zilberstein, S. 2003. Adaptive peer selection. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS) 2003*.

[7] N. Magharei, R. Rejaie, and Y. Guo, “Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches,” in *Proceedings of IEEE INFOCOM*, 2007.

[8] N. Magharei and R. Rejaie, “PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming,” in *Proceedings of IEEE INFOCOM*, 2007.

[9] M. Wang and B. Li, “Lava: A reality check of network coding in peerto-peer live streaming,” in *Proceedings of IEEE INFOCOM*, 2007.

[10] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, “Randomized decentralized broadcasting algorithms,” in *Proceedings of IEEE INFOCOM*, 2007.

[11] Bin Cheng, Lex Stein, Hai Jin, and Zheng Zhang, “Towards Cinematic Internet Video-on-Demand” *in Proceedings of Eurosys* 2008.

[12] Bin Cheng, Lex Stein, Hai Jin, and Zheng Zhang, “A framework for Lazy Replication in P2P VoD”, *in Proceeding of NOSSDAV* 2008.

[13] Bin Cheng et al. “GridCast: Improving peer sharing for P2P VoD”, *in ACM Transactions on Multimedia Computing, Communications, and Applications* (TOMCCAP). Volume 4 , Issue 4 (October 2008).

[14] Abbasi, U., Ahmed, T.: COOCHING: cooperative prefetching strategy for P2P video-on-demand system. In: Lecture Notes in Computer Science; Wired-Wireless Multimedia Networks and Services Management, vol. 5842, pp. 195–200. Springer, Berlin (2009).

[15] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, “The feasibility of supporting large-scale live streaming applications with dynamic application end-points,” in *Proceedings of ACM SIGCOMM*, Portland, OR, USA, Aug. 2004, pp. 107–120.

[16] K. Sripanidkulchai, B. Maggs, and H. Zhang, “An analysis of live streaming workloads on the internet,” in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA,2004, pp. 41–54.

[17] E. Veloso, V. Almeida, W. M. Jr., A. Bestavros, and S. Jin, “A hierarchical characterization of a live streaming media workload,” *IEEE/ACM Transactions on Networking*, vol. 14, pp. 133–146, Feb. 2006.

[18] R. Kumar, Y. Liu, and K. Ross, “Stochastic fluid theory for p2p streaming systems,” in *Proceedings of IEEE INFOCOM*, 2007.