On Supporting P2P-based VoD Services over Mesh Overlay Networks

Mostafa Fouda*[1], Tarik Taleb†[2], Mohsen Guizani‡[3], Yoshiaki Nemoto*[4], and Nei Kato*[5].
*Graduate School of Information Sciences (GSIS), Tohoku University, Japan.
†NEC Europe Ltd.
‡Department of Computer Science, Western Michigan University, USA.
Emails: mfouda@it.ecei.tohoku.ac.jp[1], tarik.taleb@nw.neclab.eu[2], mguizani@ieee.org[3],
nemoto@nemoto.ecei.tohoku.ac.jp[4], and kato@it.ecei.tohoku.ac.jp[5]

***Abstract*—Due to their ability to overcome many shortcomings associated with the contemporary client-server paradigm, Peer-to-Peer (P2P) networks have attracted phenomenal interests from researchers in both academia and industry. Interactive and multi-media streaming applications using P2P networks are, however, often prone to long startup delays, which disrupt the smooth playback and undermine users' perceived quality of service. In addition, P2P networks must be able to support a potential number of users while ensuring that the resources are efficiently utilized. In this paper, by addressing these shortcomings in the traditional P2P framework, we envision a novel scheme to effectively provide a Video-on-Demand (VoD) using P2P-based mesh overlay networks. The proposed scheme covers two main phases, namely requesting and scheduling modes. The former aims at dynamically selecting the required contents from the available peers. On the other hand, in the scheduling mode, the incoming requests are scheduled in a priority-based manner for minimizing the startup latency and sustaining the playback rate to an acceptable level. Computer simulations have been conducted to verify the effectiveness of the proposed scheme. The obtained results demonstrate the scalability of our envisioned scheme in addition to its capability to reduce the startup delay and provide a sustainable playback rate.**

I. Introduction

Peer-to-peer (P2P) based approaches have enjoyed a great success over the recent years for distributing contents over the Internet. This is, indeed, evident through the ever-increasing popularity of numerous P2P applications such as BitTorrent [1], Emule [2], SopCast [3], and PPLive [4]. The diversity in the services offered by these P2P systems is remarkable. A common yet popular application in P2P-oriented networks is file-downloading, e.g., by employing BitTorrent [1] and Emule [2] programs. A user (i.e., a peer) typically shares a file in such systems that can be downloaded by interested users (i.e., other peers). The prime objective of this approach is to eliminate the need to have a server to contain and distribute this file. Indeed, through P2P-based networks, the peer that owns a file can help one or several interested peers to start downloading the file followed by all the peers helping each other to accelerate the download process at decent rates. The evolution of the Internet has, however, led to the emergence of more sophisticated applications (e.g., live streaming video services). By employing the P2P approach, some research work such as that in [5] considered streaming live videos to subscribers in a P2P fashion. Commercial systems such as PPLive [4] have also adopted P2P-based live video streaming techniques. The benefit of employing P2P technologies in streaming live video feeds lies beneath the fact that while many users are watching a certain video, these users can collaborate with one another (by sharing the pieces of the video that they already have) to reduce the streaming load on the server. However, the main challenge that pertains to this technique is to formulate a system, which ensures that all the involved peer-entities can successfully receive the streamed content and play the same smoothly (i.e., at decent rates).

A variant of the streaming applications has recently emerged in the form of P2P Video-on-Demand (P2P-VoD) services. Huang *et al.* [6] demonstrated that P2P-VoD systems can reduce the streaming servers' load (e.g., Youtube [7] content servers) significantly, thereby saving money (in the order of millions of dollars) involved in deploying servers with replicated contents and maintaining high-bandwidth links. As a matter of fact, these P2P-VoD systems, similar to their live streaming counterparts, deliver the contents via streaming. The main difference between these two strategies lies in the fact that peers in a P2P-VoD framework are able to watch different segments of the same video simultaneously. This impacts the peers' ability to help one another in serving the streamed video content as efficiently as it would have been possible in case of live streaming scenarios.

As a remedy to the above issue, the P2P-VoD systems, rather than solely employing the playback buffer (with limited memory) of the peer-terminals, often impose upon its peers to contribute higher disk storage. To facilitate such a framework, it is imperative to design scheduling mechanisms capable of instructing the involved peers to assist one another in streaming the VoD contents in a real-time manner. To this end, we propose a service scheme to efficiently provide a Video-on-Demand (VoD) scheme over P2P mesh-based overlay networks with user-scalability issues in mind. The proposed scheme consists of two main phases, namely requesting and scheduling modes. In the requesting mode, the main idea is to dynamically select the required contents from the available peers. In the latter, the inbound requests are scheduled in a priority-based manner in order to minimize the startup latency and sustain the playback rate. Some simulations have been conducted to verify the applicability of the proposed system. The obtained results demonstrate the proposed scheme's ability to reduce the startup delay and to provide a sustainable playback rate.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

The rest of this paper is organized as follows. Section II discusses some related work while Section III presents our proposed system. The conducted simulation results and analyses are presented in Section IV to verify the performance of the proposed system. Finally, the paper concludes in Section V.

II. Related Work

In recent years, a large number of research work has focused on exploring adequate streaming strategies to enhance the streaming quality and also to meet the requirements of the multimedia streaming over the Internet. One particular problem in the P2P live streaming domains noted by many researchers is the scheduling problem [8], [9], [10], [11]. However, few work have been conducted to study its impact on the P2P-VoD systems [12], [13], [14], which are more difficult to design and implement. There are significant differences between live streaming and VoD streaming technologies. For example, users' interactive behaviors such as pausing and random jumping are allowed when they subscribe to VoD services. In contrast, live streaming systems do not provide these features to the subscribers. As a consequence, the design and deployment of a real-time P2P-VoD system become incredibly difficult compared to those of a P2P live streaming system.

In order to develop scalable VoD services, Neighbors-Buffering Based VoD (NBB-VoD) [15] scheme was proposed. This scheme utilizes both multicast and P2P technologies. The key idea behind NBB-VoD is to deliver video contents over staggered multicast channels. If a request for a particular video item comes in between the start times of two consecutive channels, the missing portion of the video is streamed from the buffering of neighbors in a P2P fashion. While NBB-VoD exhibits reasonably low start-up delays, it relies on a multicast-enabled framework. Unfortunately, the IP-multicast-capable networks are yet to be widely deployed since they require special routers.

Contemporary researchers also attempted to combine P2P techniques with the server-client streaming model to build a hybrid system called BitTorrent-Assisted Streaming System (BASS) [16]. The architecture is composed of two components, namely an external media server and a modified BitTorrent [1] protocol. The only modification made to the BitTorrent protocol is that the clients are not enabled to download any data prior to the current playback time. The BASS clients download chunks from the media server sequentially. However, they do not need to acquire, from the server, the chunks which are already/being downloaded by the BitTorrent. Even though the BASS system reduces the load on the server (compared with the server-only scenario), the server load still linearly increases with the growing number of users which obviously restricts the scalability of the system.

To alleviate the above problem, Vlavianos *et al.* [12] did not consider any external media server and proposed the Bitorrent Streaming (BiToS) strategy with the ability to support VoD streaming based on the BitTorrent [1] protocol. They introduced a piece selection mechanism to replace the "rarest-first" technique used in the original BitTorrent approach. However, their work considers the chunk, which cannot meet its playback time, as a missing chunk. The missing chunks are assumed not to be downloaded and therefore, they cannot be played back. This hampers smooth playback of the video stream and consequently affects the quality of experience as perceived by the users.

In order to efficiently deploy P2P-VoD systems, some researches focused on providing small startup latencies. The work in [13], for example, combines network coding with segment scheduling to achieve a high utilization of the system resources. However, this approach does not take into account realistic parameters pertaining to practically deployable P2P-VoD topologies.

The above research work lead us to formulating the problem scope pertaining to P2P-VoD mechanisms, particularly in the mesh overlay networks. In a mesh overlay network, each node contacts a subset of neighbors to receive a number of chunks. Each node needs to know which chunks are owned by its neighbors and explicitly pulls the chunks it needs. In essence, every node relies on multiple neighbors to retrieve the content which makes the system resilient to node-failures. It is worth noting that a peer may simultaneously request multiple neighbors in two ways, namely for either the same content or for different ones [14]. The former approach attempts to receive the content within due time, but leads to many replicated transmissions. On the other hand, the latter approach may fail to retrieve, from the busy neighbors, the chunks, which are more urgently needed to play back the content smoothly within due deadlines.

To address the above issues, we carefully redesign the requesting mechanism of a peer to its neighbors by implementing a four-way handshake mechanism to (i) avoid duplicate transmissions and (ii) retrieve the chunks that are more urgent for smooth streaming. In brief, our contributions are three-fold, namely envisioning an efficient VoD system over P2P mesh overlays, developing a prototype to test the feasibility and effectiveness of the envisioned approach, and finally evaluating its performance. The envisioned VoD system comprises a tracker selection mechanism along with adequately designed requesting and scheduling algorithms implemented in the downstream and upstream peers, respectively.

III. The Proposed Scheme

The previous sections revealed that more attention needs to be paid to designing an efficient P2P-VoD system. In this section, we present our proposed system, which hinges on its ability to efficiently utilize the overlay network resources (the available upload bandwidth of the content source node and that of each peer). Our system can be described as follows.

A. System Basics

The system is composed of one source node and a set of N peers. The peers are interested in some video content, which initially exists on the source node. The peers have the ability to join the system at any random point of time, and they

978-1-4244-4148-8/09/$25.00 ©2009

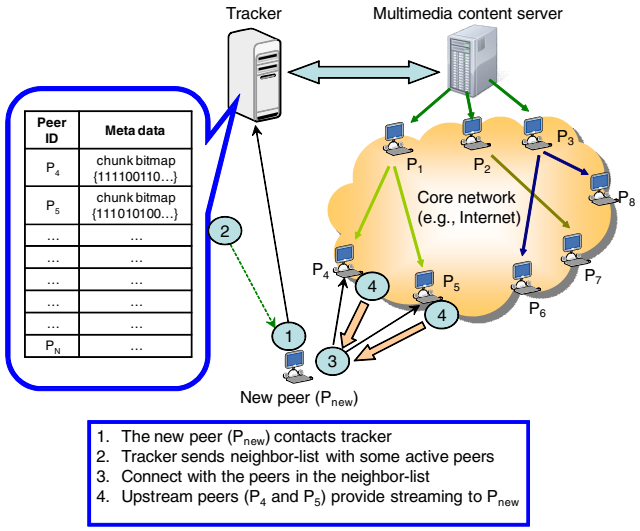

Fig. 1. The mechanism of a new host/peer joining the P2P-VoD overlay.

can watch the video sequentially from the beginning or from anywhere through the video file. The resources (especially network bandwidth) of the source node are limited, and hence, peers should contribute with their own resources to the delivery of the contents. The upload and download capacities of the peers are limited, heterogeneous, and typically asymmetric.

The video content is divided into small chunks of fixed size. The system is "media codec agnostic", i.e., it may be used with any codec stored in an ISO-based media file format and is, therefore, not constrained to MPEG-4 codecs alone. As a consequence, the subscribing nodes need to download all the chunks of the video file; if a chunk is not available when required, the video freezes and this results in a rather undesirable effect on the smoothness of the video playback. Peers are assumed to have enough storage space to keep all the already downloaded chunks. Peers that are receiving the stream will be referred to as downstream peers. On the other hand, the peers that are providing the stream to other nodes are called upstream peers throughout the paper. It should be stressed that a peer may act as an upstream and also as a downstream peer at the same time.

In addition, a central tracker is responsible for providing the existing as well as the new peers with an updated list of their neighbor active peers, referred to as the neighbor-list. This list will be chosen by the tracker according to certain criteria, which will be delineated later.

Fig. 1 demonstrates the basic architecture for a tracker-based P2P mesh overlay network. In this architecture, a newly joining peer contacts the tracker to get a neighbor-list containing some active peers. Upon receiving the neighbor-list, the new peer immediately starts requesting the video chunks from the peers available in the neighbor-list. It is also worth mentioning that a peer, at any point of time, is connected with a small subset of active peers, and may exchange contents and control messages only with them.

Each peer also regularly sends keep-alive messages to the tracker to report its chunk bitmap and other statistics. This information is collected by the tracker for monitoring and management purposes. Once such information is acquired from the available peers, the tracker automatically updates and resends the neighbor-lists to the downstream peers.

B. Tracker Function

The tracker can be considered as a directory server. It maintains a database of peers' information. In addition, it carries out a number of important calculations during the phase of building the neighbor-list for downstream peers. The tracker is supposed to collect and save the information pertaining to each peer, e.g., its IP address, port number, chunk bitmap, and upload capacity. The tracker sends a neighbor-list to a downstream peer in either of the following ways.

i. Periodically: if the current neighbor-list requires modification.
ii. On demand: in case a new peer joins the service, or if an existing peer wants to change the playback time of its video stream (e.g., forward or backward).

In our proposed scheme, the tracker does not arbitrarily select the peers that will be incorporated in the neighbor-list and which will be later sent to an appropriate downstream peer. The proposed peer selection mechanism for constructing a neighbor-list for a downstream peer follows four steps, namely (i) the selected peers need to have the chunks following to the downstream peer's last playable chunk, (ii) the number of times a peer appears in all the neighbor-lists is proportional to its upload capacity, (iii) the source node is not to be included in the neighbor-list if the number of already chosen neighbors is sufficient, and (iv) the tracker may assign randomly chosen peers to fill out the neighbor-list if it is incomplete even with the inclusion of the source node.

The first step is important in a way that if the tracker randomly selects the upstream peers, it may select a peer which is still watching an earlier part of the movie and does not have the requested chunks. The aim of the second step is to distribute the load over the peers according to their resources and not to overload any of them which may lead to instability in the VoD delivery service. The third step is aimed at using the source node in only critical situations since it always possesses all the chunks. Nevertheless, if there are enough upstream peers to fill out the neighbor-list and serve the downstream peer, it is certainly better to save resources of the source node. The fourth and final step aids the downstream peers to increase the chances of utilizing their own download bandwidths efficiently, as these randomly selected peers may probably possess some missing (although not urgently required) chunks.

C. Downstream/Upstream Communication Algorithm

Once the tracker assigns a group of upstream peers (in the form of a neighbor-list) to each downstream peer, the downstream peer starts to communicate with those peers in order to receive its missing chunks from them. At this point, evaluating the exact number of chunks and which chunks should be requested from a given peer from within the neighbor-list are,

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

indeed, challenging tasks and they need to be addressed in an efficient manner. To deal with this issue, we propose a novel algorithm, which consists of a four-way handshake mechanism and three modules. Two of the modules are executed by the downstream peers while the third one is carried out by the upstream ones. The details of the algorithm are provided as follows.

1. Downstream peer's requesting module part i: As illustrated in Fig. 2, a downstream peer implements the first module in our algorithm. It is responsible for counting the amount of chunks to be requested from each upstream peer based on the received information about each upstream peer in the neighbor-list. Such information includes the chunk bitmap of each upstream peer and hence, the downstream peer will be able to know which upstream peer has the required chunks and also the maximum number of required chunks that exist in the storage media of that upstream peer. The downstream peer will then request a specific number of bandwidth slots from each upstream peer where each slot is enough to transfer an individual chunk. The number of these slots is directly proportional to the upstream peer's upload capacity. In fact, linking between the number of slots that will be requested from each upstream peer with its upload capacity will result in enhancing the utilization efficiency of the available bandwidth. In addition, it will also increase the probability of approving such amount of requested slots in the upstream peers' end.

We define a new parameter, Time-To-Freeze (TTF), as the remaining time left for the video playback to go to the freeze state. A simple example is shown in Fig. 3. While requesting the upstream peers, the downstream peer sends the number of required slots along with its own Time-To-Freeze (TTF). The upstream peer uses this TTF during the scheduling process for prioritizing requests.

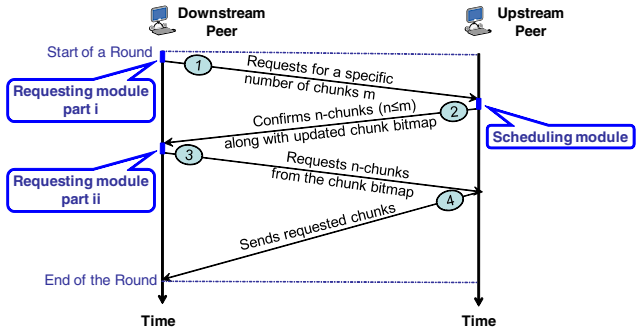

Fig. 2. The proposed four-way handshake mechanism.

2. Upstream peer's scheduling module: When some downstream peers request a certain number of bandwidth slots from an upstream peer, the latter prioritizes these requests in such a manner as to achieve optimum performance via an efficient distribution of its resources. In our scheduling algorithm, the highest priority will be assigned to the downstream peer that has the minimum TTF to prevent playback freeze (i.e., to achieve smooth playback) which is one of the main targets in the proposed system. The second prioritization will be carried out according to the upload capacity of the peers. This is to provide the higher upload capacity peers with more chunks so that they may utilize their upload capacities efficiently while distributing the chunks to other downstream peers. This can be summarized by the following rules:

Rule 1: The upstream peer assigns up to $K\%$ of its available upload bandwidth to the downstream peers that have ($TTF < TTF_{th}$), where TTF_{th} is a threshold. The peer with the lowest TTF will be assigned the highest bandwidth.

Rule 2: The upstream peer distributes the remaining available upload bandwidth among the requesting downstream peers according to their upload capacities. Peers with higher upload capacities will be assigned more bandwidth.

It should be noted that a newly joining peer has not yet received any chunk and therefore, its Time-To-Freeze (TTF) is zero, which gives its request the highest priority. This decreases the startup delay of its playback to a minimum level.

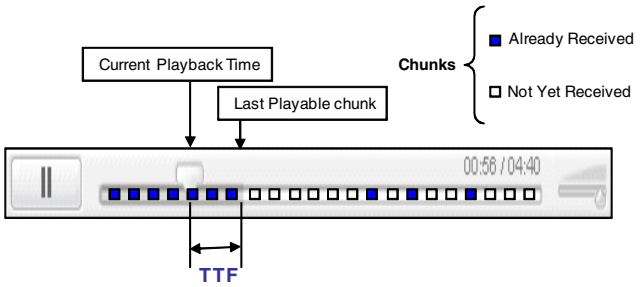

Fig. 3. Illustrating TTF in a simple example.

3. Downstream peer's requesting module part ii: This module can be also referred to as a chunk selection mechanism, which is another vital part in our algorithm. Following the upstream peers' replies with the number of approved bandwidth slots that have been assigned to the downstream peer, the chunk selection algorithm is triggered. The main policy of this part is to efficiently choose the chunk numbers that will be requested from each upstream peer according to the number of approved slots. Since the main aim of the system is to maintain smooth playback, the priority in requesting chunks is given towards those particular chunks, which appear just after the playback time with a maximum interval equal to a predefined time window, T. After all the chunks in T have been requested, the downstream peer may consider that the streaming quality will not be disrupted during the next T time window. Therefore, there is no need to request chunks sequentially. On the contrary, at this stage, if the downstream peer can ask for more chunks from its corresponding upstream peers, it may consider requesting chunks randomly to increase the chunks diversity in the overlay.

IV. Performance Evaluation

In this section, the simulation model is first described. The definition of the performance metrics used in evaluating the system is followed. The algorithms used in the comparison is then defined. Finally, the performance of the envisioned approach is discussed.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

A. Simulation Set-up

In order to evaluate our approach, we have developed a new simulator model based on the one proposed in [13] with different scenarios. We use a pool of 200 heterogeneous peers (i.e., peers with different capacities and resources). The simulator operates in discrete intervals of time called rounds. At every round, each node contacts its neighbors to request and receive a number of video chunks. All chunk transfers, both between peers and from the source node, occur simultaneously, and then the system moves to the next round.

B. Performance Metrics

The main performance metrics, which we consider to evaluate the performance of the envisioned peer-to-peer streaming system, are as follows.

1. Maximum supportable playback rate: During the video playback, chunks have to be received prior to the playback deadline. The chunks missing the deadlines cause the video playback to freeze awaiting their arrival. This degrades the quality of service perceived by the users. The maximum supportable playback rate, given the startup delay (or initial buffering), is the maximum rate at which the video can be played without being frozen.

2. Peer's bandwidth utilization: The bandwidth utilization reflects how effectively the envisaged algorithm utilizes the bandwidth resources. A poor scheduling scheme may underutilize a peer's available upload bandwidth while other peers are "starved" of contents exhibiting the so-called *content bottleneck* phenomenon. The bandwidth utilization is computed by estimating the average upload rate and dividing it by the peers' pre-set upload capacity.

Following these two performance metrics, Annapureddy *et al.* mentioned, in their work [13], two naive approaches, namely the random and sequential algorithms, which we use as comparison terms. In the former, the downstream peers request chunks in a random order from arbitrarily selected neighbor peers that own those chunks. This strategy results in high chunks diversity and decent system throughput. However, nodes receive chunks in an out-of-order manner that may not be useful to sustain a good playback rate. On the other hand, in the second approach, as the peers consume the chunks of the video sequentially, the downstream peers prefer to request the chunks that are closest to what is needed for the video playback. This strategy results in a better playback rate than the random algorithm. However, due to the fact that the peers contain rather similar chunks, there are fewer chances to find and exchange diverse chunks. This leads to reduced throughput in the sequential algorithm in contrast with the approach that randomly chooses the chunks.

C. Results and Discussion

We perform computer simulations to evaluate the performance of VoD over mesh P2P overlay networks. Given a content source node and a set of peers with known upload capacities, the maximum/optimal streaming rate, R_{max}, can be formulated from the average upload capacity per peer as follows:

$$R_{max} = \frac{u_s + \sum_{i=1}^{N} u_i}{N} \quad (1)$$

where u_s, u_i, and N denote the upload capacity of the media source node, the upload capacity of peer i, and the number of peers in the system, respectively.

In our simulation, we address the heterogeneity due to the different bandwidth of the access network. In order to do this, the download and upload capacities of the considered peers are set according to the bandwidth distribution used in [17], which is derived from the actual distribution of Gnutella nodes. Specifically, 20% of the peers have upload capacity of 128 kbps, 40% have 384 kbps, 25% use 1 Mbps, and 15% employ 5 Mbps as shown in Table I. The upload capacity of the source node is set to 5Mbps during the simulation. By using Eq. 1, we found the maximum/optimal streaming rate of the system, R_{max}, to be 1280Kbps for 200 peers.

The simulation parameters are listed in Table II. We have tested the worst case in a VoD streaming system over P2P overlay which is the flash crowd scenario. In this scenario, 200 peers simultaneously join the stream and each of these peers attempt to watch the movie from the beginning, while only one source node has the video content.

We conduct the simulation by setting different values of K (the maximum bandwidth percentage that the upstream peer assigns for peers that have TTF below TTF_{th}). From the simulation results, it follows that the best value of K for this network setting is found to be 40%. The scenario where K=0% is also included in the results in order to demonstrate the difference in performance in these two conditions. It should be noted that K=0% implies that the upstream peers assign all the upload bandwidth only according to the available upload capacity of the downstream peers without taking the TTF values into consideration.

Owing to space limitation, we present, in Fig. 4, the maximum supportable playback rates for different startup latencies for only peers of Class B. The reason behind our choice is the

TABLE I
PEER CLASSES

Peer's Class	Percent of Peers	Download BW	Upload BW
Class A	15%	10 Mbps	5 Mbps
Class B	25%	3 Mbps	1 Mbps
Class C	40%	1.5 Mbps	384 Kbps
Class D	20%	784 Kbps	128 Kbps

TABLE II
SIMULATION PARAMETERS

Simulation Period	1000s
No. of Peers (N)	200
Video File Size	16 MByte
Chunk Size	8 KByte
No. of Neighboring Peers	8
Tracker Update Period	10s
TTF_{th}	20s
T	20s

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

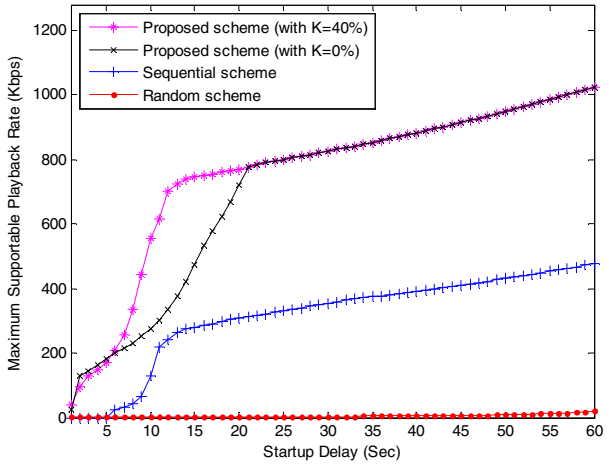

Fig. 4. The maximum supportable playback rates for different startup delays in case of Class B peers.

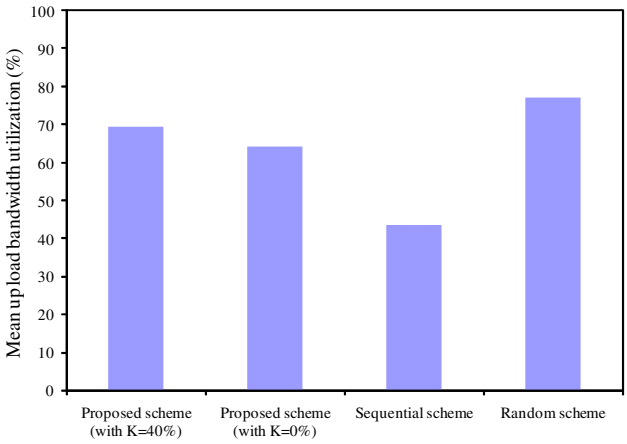

Fig. 5. Mean upload bandwidth utilization in the considered schemes.

fact that class B peers contribute to a reasonable amount of upload/download bandwidth in the overall system (i.e., they are not as extremely distributed as the peers belonging to the other classes). As depicted in the figure, the proposed scheme with K=40% exhibits much better performance compared to the other approaches in terms of each peer's ability to play the movie at a good rate while maintaining substantially low startup latency. Furthermore, for the lower startup delays up to 20s, the TTF values of the peers are also low, and K=40% helps them to be serviced with necessary chunks for performing smooth playback.

In Fig. 5, we plot the mean upload bandwidth utilization of the participating nodes in the system during the 40^{th} to 80^{th} seconds of each simulation run. Of course, the naive random algorithm has higher utilization of resources because of the chunks diversity. However, this does not assist the concerned peer in playing the movie smoothly. On the other hand, our proposed approach, in particular with K=40%, achieves significantly high bandwidth utilization (under the flash crowd scenario) which is about 70% of the aggregate upload bandwidth.

V. Conclusion

In this paper, we addressed the streaming of VoD multimedia contents through the Internet by using mesh P2P overlays. In order to achieve effective P2P-VoD streaming, it is imperative to assure a number of conditions, namely short startup latency, smooth playback, and scalability in terms of a high number of users. These conditions may be highly affected by upload bandwidth utilization of the involved peers. We described, in detail, the mechanism in which the peers can communicate with one another to decide which chunks are to be requested. In addition, a scheduling algorithm at the upstream peer end (based on its available upload capacity) is proposed. The effectiveness of the proposed scheme is demonstrated via simulations. The system exhibits encouraging performance even under worst case scenarios whereby a large population of flash crowd appears and influences the considered network. Our next step will be to study how to deal with unsynchronized requests from multiple downstream peers.

References

[1] "BitTorrent," http://www.bittorent.com.
[2] "Emule," http://www.emule.com.
[3] "SopCast," http://www.sopcast.com.
[4] "PPLive," http://www.pplive.com.
[5] X. Zhang, J. Liu, B. Li, and T. S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient live media streaming," in *Proc. IEEE INFOCOM'05,* Miami, FL, USA, Mar. 2005.
[6] C. Huang, J. Li, and K. W. Ross, "Can Internet video-on-demand be profitable?," in *Proc. ACM SIGCOMM'07,* Kyoto, Japan, Aug. 2007.
[7] Youtube, "Youtube Homepage," http://www.youtube.com.
[8] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. IEEE INFOCOM'07,* Anchorage, Alaska, USA, May 2007.
[9] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *Proc. IEEE INFOCOM'07,* Anchorage, Alaska, USA, May 2007.
[10] Y. Guo, C. Liang, and Y. Liu, "Adaptive Queue-based Chunk Scheduling for P2P Live Streaming," in *Proc. International Federation for Information Processing (IFIP),* Singapore, May 2008.
[11] C. Liang, Y. Guo, and Y. Liu, "Is Random Scheduling Sufficient in P2P Video Streaming?," in *Proc. 28^{th} International Conference on Distributed Computing Systems (ICDCS'08),* Jun. 2008.
[12] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in *Proc. IEEE Global Internet,* Barcelona, Spain, Apr. 2006.
[13] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez, "Exploring VoD in P2P Swarming Systems," in *Proc. IEEE INFOCOM'07,* Anchorage, Alaska, USA, May 2007.
[14] Y. Huang, Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-scale P2P-VoD System," in *Proc. ACM SIGCOMM'08,* New York, NY, USA, Oct. 2008.
[15] T. Taleb, N. Kato, and Y. Nemoto, "Neighbors-buffering-based video-on-demand architecture," J. Signal Processing: Image Communication, Vol. 18, No. 7, Aug. 2003, pp. 515-526.
[16] C. Dana, D. Li, D. Harrison, and C. Chuah, "BASS: BitTorrent assisted streaming system for video-on-demand," in *Proc. International Workshop on Multimedia Signal Processing (MMSP),* Shanghai, China, Nov. 2005.
[17] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Network Performance Mechanisms," in *Proc. IEEE INFOCOM'06,* Barcelona, Catalunya, Spain, Apr. 2006.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.