Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Is playing-as-downloading feasible in an eMule P2P file sharing system?*

Wen-yi WANG†, Yao-wu CHEN

(*Advanced Digital Technology and Instruments Institute, Zhejiang University, Hangzhou 310027, China*)

†E-mail: walker_wwy@hotmail.com

Received July 5, 2009; Revision accepted Aug. 4, 2009; Crosschecked May 4, 2010

Abstract: Peer-to-peer (P2P) swarm technologies have been shown to be very efficient for large scale content distribution systems, such as the well-known BitTorrent and eMule applications. However, these systems have been designed for generic file sharing with little consideration of media streaming support, and the user cannot start a movie playback before it is completely downloaded. The playing-as-downloading capability would be particularly useful for a downloading peer to evaluate if a movie is valuable to be downloaded, and it could also help the P2P content distribution system to locate and eliminate the polluted contents. In this paper we address this issue by introducing a new algorithm, wish driven chunk distribution (WDCD), which enables the P2P file sharing system to support the video-on-demand (VOD) function while keeping the P2P native downloading speed. A new parameter named next-play-frequency is added to the content chunk to strike a replication balance between downloading and streaming requests. We modify the eMule as the test bed by adding the WDCD algorithm and then verify the prototype implementation by experiments. The experimental results show that the proposed algorithm can keep the high downloading throughput performance of the eMule system with a good playing-as-downloading function.

Key words: Peer-to-peer (P2P), Video-on-demand (VOD), Playing-as-downloading, eMule
doi:10.1631/jzus.C0910408 **Document code:** A **CLC number:** TN919.8

1 Introduction

Over the last decade, the Internet Protocol (IP) based video-on-demand (VOD) applications have grown tremendously and have become the killer applications. The end user can start a movie playback without waiting to download the full video by such VOD applications as the famous Web 2.0 video site of YouTube (http://www.youtube.com). These systems provide a good playback quality and a large scale support for the video content distribution; however, they normally are closed systems with proprietary protocols and architectural design. These centralized VOD systems are designed and implemented as the server-client mode. The operator needs a great deal of money and effort to run and maintain these systems allowing the user base to continue to grow (Liu Y *et al*., 2008), owing to the bottleneck of (1) the server's disk I/O speeds and (2) the server's network bandwidth (Ghose and Kim, 2000).

Compared to centralized server-client mode content distribution systems, peer-to-peer (P2P) based content sharing systems are emerging as cheaper and more flexible solutions as every peer contributes its storage and uploading bandwidth while receiving data. Since the first P2P file sharing application Napster (Androutsellis-Theotokis and Spinellis, 2004) was created, several successful P2P applications have been deployed in the Internet widely, e.g., BitTorrent (Pouwelse *et al*., 2005) and eMule (Kulbak and Bickson, 2005). With these applications and technologies, the user can share and exchange media contents in a P2P way without any authorization from an administrative site or organization. The P2P

* Project (No. 2005C11001-02) supported by the Science and Technology Plan of Zhejiang Province, China

architecture also eliminates the bandwidth and storage requirements for the central server because all contents are stored in the peer's storage. Both eMule and BitTorrent have proven to be very effective P2P content distribution mechanisms (Vlavianos *et al*., 2006); however, they do not have the streaming support as the traditional VOD systems (Rodriguez *et al*., 2006). This limitation is a known design issue of the P2P file sharing applications. These P2P overlay networks are mesh based systems and they obtain the high swarming (downloading) efficiency by replicating the chunks randomly with the rarest first order (Füllemann, 2005; Koo *et al*., 2005; Bickson *et al*., 2007). However, a streaming application requires the chunks arrive at the receiving peer in sequential order. So the challenge here is how we can add the playing-as-downloading (PAD) service to the P2P file sharing system without impacting its high downloading throughput.

Although some of the recent systems such as BitTos (Vlavianos *et al*., 2006) and Bass (Dana *et al*., 2005) have added streaming support for BitTorrent, it is still an open issue whether similar P2P technology can be used for VOD applications. In this paper, we investigate how to enhance the streaming support for eMule. As far as we know, this is the first time this topic has been studied. We propose an enhancement algorithm named wish driven chunk distribution (WDCD) based on the original eMule chunks selection and replication algorithm. The new algorithm enables the eMule clients to have the playing-as-downloading capability, thus improving user experience by less startup time to view a movie, especially when the movie is polluted or of poor quality. Thus, end users can protect the resource utilization by canceling the downloading request at an earlier stage. We keep the eMule framework and its protocol and extend its chunk selection algorithm by adding a new next-play-frequency parameter to the chunk, indicating how urgent it is for the streaming clients. By balancing the chunk replication priority between streaming and downloading clients, our proposal enables the chunks to arrive at the receiving clients with (1) the playing order to have the smooth playback and (2) the rarest first order to have the maximum downloading throughput.

2 Related works

The P2P paradigm has been used widely for VOD systems since it was created for file sharing applications such as eMule, BitTorrent, Kazza (Leibowitz *et al*., 2003), GnuTella (GnuTella, 2009), and Freenet (Sandberg and Wiley, 2000). The P2P VOD service is more challenging than the P2P file sharing system because it should allow users to watch the movie with a short startup time and small playback jitters. The receiving peers in a P2P VOD system should obtain the chunks from different parent nodes in playback order, which will impact the efficiency of the rarest first swarming protocols used in P2P file sharing applications.

P2P VOD systems can be divided into two categories according to the overlay network structure (Liao *et al*., 2007; Magharei *et al*., 2007; Liu JC *et al*., 2008; Liu Y *et al*., 2008): tree based and mesh based. Tree based systems use a push method to disseminate the data over the application level multicast (ALM) trees (Jannotti *et al*., 2000; Chu *et al*., 2002; Castro *et al*., 2003; Tran *et al*., 2003; Guo *et al*., 2003; 2008; Do *et al*., 2004; 2008; Lee *et al*., 2005). The ALM tree structure uses the same idea as the multicast technology, so it is easy to implement and maintain. There are two kinds of ALM tree overlay structures: one is the single ALM tree, such as DirectStream (Guo *et al*., 2008), OverCast (Jannotti *et al*., 2000), ESM (Chu *et al*., 2002), and ZIGZAG (Tran *et al*., 2003); the other is the multiple ALM trees, such as SplitStream (Castro *et al*., 2003), Bullet (Kostic *et al*., 2003), P2VOD (Do *et al*., 2004; 2008), and P2CAST (Guo *et al*., 2003). In ESM, the peers join in a spanning tree and the data are forwarded from the root node to the descendant nodes along the tree. Thus, the server loading is reduced and the system service scale is improved. DirectStream creates an ALM tree at the server for every stream. When a new peer joins in the stream's ALM tree, it will search all the nodes at the indexing directory server for the best suitable one as the data forwarding parent node. ZIGZAG divides the peers into different logical layers, and then generates different clusters among the different logical layers. This design restricts the ALM tree's width and depth and thus shorten the transfer delay between the root node and the leaf nodes. The single ALM tree structure is

vulnerable to peers churn. When a peer leaves the ALM tree, all its descendant nodes must take some time to find the new parent to fix the ALM tree. This tree fixing process will obviously increase these nodes' playback jitters. Because of the heterogeneity of P2P network nodes, the end-to-end transfer delay may sometimes be large, especially when a peer with low uploading bandwidth is in the data forwarding path. The multiple ALM trees structure extends the source node number of the receiving peer to fully utilize the network resources, and thus it has stronger system stability. SplitStream (Castro *et al*., 2003) uses the multiple description coding (MDC) technology to encode and transfer the streams. Every MDC stream uses an ALM tree to forward the data, and the tree node at one of the ALM trees can also serve as the parent node for the other ALM tree nodes, so the uploading bandwidth of every tree's leaf nodes is fully utilized. P2VOD designs a fixed length cache buffer for every peer and defines the R-Block as the smallest video data unit. The peers with the same smallest R-Block number are grouped together and defined as a video session. A peer at P2VOD can obtain not only the video data from the ALM tree's parent node, but also the data from the neighbors at the same video session. P2CAST uses patching technology to reduce the server loading, because a newly joining peer obtains only the video data that have missed the current playback timestamp from the server, and then receives the data after that timestamp from the other peers.

As the researches (Jiang *et al*., 2003; Dana *et al.*, 2005; Pai *et al*., 2005; Zhang *et al*., 2005; Vlavianos *et al*., 2006; Annapureddy *et al*., 2007; Liao *et al*., 2007; Magharei and Rejaie, 2007; Cheng *et al*., 2008a; 2008b; 2008c; Huang *et al*., 2008; Mol *et al*., 2008; Tian *et al*., 2008) have discussed, the peers of a mesh overlay network can obtain data from multiple source peers and forward data to multiple child peers compared with the tree based structure, so the mesh structure can support larger scale systems and are more tolerant to peers churn. CoolStreaming (Zhang *et al*., 2005) defines a member list for every peer. The receiving peer generates a partner list from his/her member list, receives data from his/her partners, and runs a scheduler to update the member and partner lists periodically to select the most suitable parent node dynamically. GridCast (Cheng *et al*., 2008a; 2008b; 2008c) and PPLive (Huang *et al*., 2008) cache one or multiple files at the peers to improve the VOD performance. AnySee (Liao *et al*., 2007) divides the peers into two sets, the active set and the backup set. The receiving peer calculates periodically the smallest transfer path between itself and all the other peers who have cached the same data, and then updates the peer of the smallest delay path to the backup set. When one of the paths at the active set is disconnected, the receiving peer can build up very quickly a shortest data forwarding path from the backup set to avoid the playback jitters. Prime (Magharei and Rejaie, 2007) designs a two-phase data distribution algorithm. It uses the ALM trees to distribute the MDC streams in the diffusion phase. Once the diffusion phase is completed, the streams are forwarded using swarm technology and one diffusion ALM tree node can obtain the other layer's MDC stream data from the same or deeper degree nodes of another diffusion ALM tree. This improves the bandwidth utilization of the tree's leaf nodes. The peers at the mesh overlay network select the source nodes randomly and globally, so they normally will have longer startup time. They often need very large cache buffer for a good VOD playback performance, such as the GridCast and PPLive, which cache the whole file at the peers like a P2P file sharing system.

In some of the recent researches, besides using the P2P paradigm to improve the VOD system performance, how to add streaming capability to the P2P file sharing system is also studied; e.g., Bass adds a streaming media server to the BitTorrent network for streaming support. The playing client obtains the chunks before the playback deadline from the media server and fetches the chunks after the playback deadline from BitTorrent peers. Based on Bass, an improved algorithm is used in BiTos. It is still based on the BitTorrent system; the difference is that the streaming media server is removed. It adds the VOD supporting by directly modifying the BitTorrent protocol. BiTos is the first research to modify the chunk replication algorithm used in the P2P file sharing application for streaming support. BiTos divides the file chunks into two sets, the high priority set and the low priority set. The high priority set contains the chunks that are close to the playback deadline and

have not yet been downloaded; the low priority set contains the remaining chunks that have not been downloaded and are further away from the playback deadline. A selection process is used to decide which chunk should be downloaded. A chunk in the high priority set is downloaded with probability p, and the one in the low priority set is downloaded with probability $1–p$. By setting the value of p greater than 0.5, the chunks in the high priority set will be downloaded earlier than the ones in the low priority set. Intuitively, the larger value of p offers the better chance for chunks to arrive at the receiving peer within their playback time, while a smaller value of p increases the density of chunks. Thus, it leads to a better overall downloading throughput.

In short, BiTos is the most similar to our proposed algorithm, but with several differences: (1) Our test bed is eMule, not BitTorrent. eMule and BitTorrent have totally different design at system architecture, chunk distribution, and incentive mechanism. Compared with BitTorrent, eMule implements a rough streaming function, called preview, which makes eMule more suitable to be enhanced for the streaming application. (2) Our algorithm is based on the chunk unit level and needs only a small modification to the original eMule system, while BiTos is designed on a chunk sliding window and the performance is decided by the window size. Because it is hard and complex to fine tune the parameter, BiTos still leaves it as an open issue about how to adjust the sliding window size to obtain the best performance.

3 eMule background

In current available P2P file sharing applications, eMule is the only one with limited streaming support. This feature is called preview in eMule. In this section, we present the eMule chunk selection and replication algorithm and describe how to support the preview function.

Fig. 1 shows the eMule architecture design. The eMule network is composed of hundreds of servers and millions of peer clients. The client obtains P2P services by connecting to one of the eMule servers. Although every client is pre-configured with a list of connectable servers, the client can connect to only one of them at a given time. An eMule server is running a central index service to store the connected clients information and chunks information of their shared files. eMule is a receiver driven design, so it is up to the downloading client to select the chunks for replication.

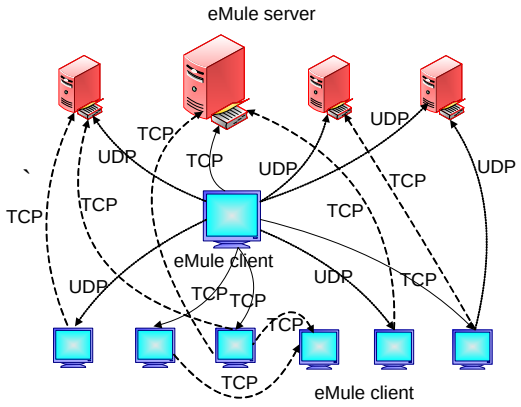

Fig. 1 eMule architecture overview

eMule is a structured P2P overlay network. Peer P_i sends the chunk information of its shared file F to server S by the TCP protocol, and another peer P_j searches out this shared file F and its source peer list from S. In the eMule system, every file is presented by a globally unique identifier (GUID). This GUID is generated by a hash function (Kulbak and Bickson, 2005). The file is cut into chunks of the same size. Like the file presentation, every chunk is also presented by a hash function generated GUID. If P_j has at least one chunk of file F, P_j is presented as the source peer of F. An eMule server uses the term of frequency to show the chunk's availability. For example, f_i equals the replicated number of chunks C_i's at the eMule peers. A larger f_i shows that C_i has a larger source peer number. Thus, a newly joining peer can download C_i more quickly.

Fig. 2 presents the eMule's chunks selection and replication algorithm. Several criteria are used to replicate as many file chunks between peers as possible according to their availability (frequency). The rarest chunks (with a lower frequency) have a higher priority to be replicated. Its design goal is to make as fast overall downloading speed as possible by providing more sources for every chunk.

To obtain the maximum overall downloading throughput, the eMule peer follows the four criteria shown in Fig. 2 to replicate the chunks.

```
procedure GetNextRequestedChunks(F)
  Define three frequency zones by bounds, veryRare-
    Bound, and rareBound;
  while Ci is F's chunk
    if (Ci is a preview block)
      criterion_preview=true;
    endif
    if (Ci is in downloading state) and (i<veryRareBound)
      criterion_requested=true;
    endif
    if (Ci has been downloaded partially)
      criterion_completion=
       downloaded size/CHUNK_SIZE;
    endif
    if (i<veryRareBound)
      Ri=i*25                                  /*Criterion 1*/
        +(criterion_preview==true)?0:1         /*Criterion 2*/
        +(100−criterion_completion);           /*Criterion 4*/
    else if (criterion_preview==true)
      Ri=(criterion_requested!=true)?10000:30000
                                               /*Criterion 3*/
        +(100−criterion_completion);           /*Criterion 4*/
    else if (i<rareBound)
      Ri=i*25                                  /*Criterion 1*/
        +(criterion_requested!=true)?10101:30101
                                               /*Criterion 3*/
        +(100−criterion_completion);           /*Criterion 4*/
    else if (criterion_requested!=true)
      Ri=20000                                 /*Criterion 3*/
        +(100−criterion_completion);           /*Criterion 4*/
    else
      Ri=40000                                 /*criterion 3*/
        +criterion_completion;                 /*criterion 4*/
    endif
  end while
  Pick the chunk with the smallest Rank value as the one to
    be the requested next;
end procedure
```

Fig. 2 eMule chunk selection algorithm

Criterion 1 The chunks with the lowest frequency (rarest) should be downloaded with the highest priority (first); it is called rarest-first. This design enables the newly joining peers to have more available sources and speeds up the downloading process.

Criterion 2 The chunks used to do preview have the second priority. In eMule implementation, the first and last chunks of a file are marked as the preview ones as players often use them to parse the correct metadata. The metadata are necessary for the players to set up and initialize the decoders.

Criterion 3 The rarest chunks that have been in the downloading process have the third priority. Although these chunks have already been downloaded at the current receiving peer, their availability is still very small as their frequency is still in the rare bound. The receiving peer tries to spread this chunk's replication request to each source peer of the file. After these source peers receive the replication request, they will put this chunk to their download request queue. This operation can increase the chunk replications in the whole network, and thus improve the overall downloading performance.

Criterion 4 Partially retrieved chunks should be completed before a peer starts to download the others.

4 The new algorithm

In this section, we describe the proposed WDCD algorithm, which adds the streaming support to the original eMule system by enhancing the chunk selection and replication algorithm. Its design goal is to make the chunks arrive at the streaming peer's player buffer within their playback time while keeping the swarming efficiency of the chunk replication algorithm among the downloading peers.

Fig. 3 gives the overview of the WDCD algorithm. The file chunks list is divided into two windows by a cursor t_{deadline} (t_{deadline} is defined in Table 1, and it is moving along with the player's wall clock of the PAD client). If one chunk downloading could not be finished within t_{deadline}, this chunk was assumed to be dropped at the PAD client player, and this will cause the player jitter and discontinuity. There are two windows in Fig. 3: (1) Buffering window. The window size is fixed and the chunk number is used as the unit. To describe the algorithm in a simple and easy way, we assume that the media file is a constant bit rate (CBR) stream and the chunk duration is fixed as one second. Thus the buffering window size has the same value as the player's buffer size in our WDCD algorithm description. (2) Next-play-window. Its window size is variable and equals the file duration minus t_{deadline}. The window size is changing when t_{deadline} is shifting, until it finally reaches zero. The chunks at this window have larger playback timestamps than t_{deadline}, so they are the next ones to be played at the player. They should be downloaded with a higher priority compared with the chunks at the buffering window. Inside the next-play-window, the

chunks with a timestamp closer to t_{deadline} have a higher downloading priority. If one chunk could not finish downloading within its playback time, it would be marked as 'missed' for this PAD client's playing request.

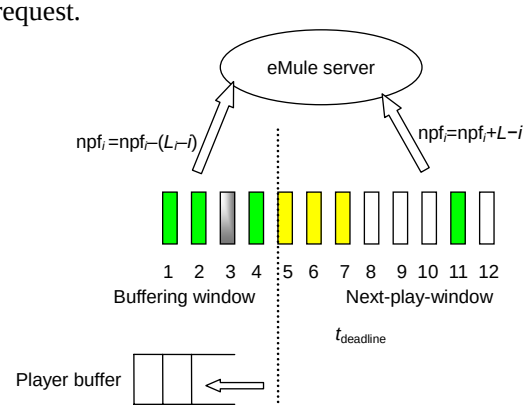

Fig. 3 The wish driven chunk distribution algorithm

Table 1 Notations used in the wish driven chunk distribution (WDCD) algorithm

Term	Definition
S	One central server of the eMule network
P_i	Peer i
C_i	The ith chunk of file F
F	One file sharing by the eMule network
f_i	The frequency of chunk C_i at the eMule network indicating this chunk's availability in eMule peers
R_i	The rank value of chunk C_i. A peer will always select the chunk with the smallest rank value to replicate
npf_i	The next-play-frequency value of chunk C_i, indicating the streaming requests density for the chunk C_i. A peer will try to replicate the chunks with higher npf_i values so that these chunks can arrive at the PAD client's player buffer before the playback deadline
L	The total chunk size of file F
t_{deadline}	The chunks playback deadline of the PAD client. If a chunk cannot reach the PAD client's player buffer within t_{deadline}, this packet is assumed to be dropped for the playing request
t_i	The timestamp of chunk C_i

In the WDCD algorithm, a new parameter named next-play-frequency (npf) (Table 1) is added to the file chunks. The chunk with a larger npf value has a higher replication priority in the peer's chunk selection procedure. The npf value of chunk C_i is calculated by

$$\text{npf}_i = \begin{cases} \text{npf}_i - (L-i), & t_i \le t_{\text{deadline}}, \\ \text{npf}_i + (L-i), & t_i > t_{\text{deadline}}. \end{cases} \quad (1)$$

In Eq. (1), the PAD client increases the npf value of the chunk that is behind the player's playback deadline. The chunk closer to t_{deadline} will have a larger npf value and it is more urgent to be replicated for streaming requests. For those chunks that have missed the playback deadline ($t_i \le t_{\text{deadline}}$), the PAD client needs to restore its npf value using Eq. (1) to indicate that it is not needed any more for this PAD client's streaming request. The main reasons for the condition $t_i \le t_{\text{deadline}}$ branch of Eq. (1) are:

1. For the chunks that have been missed at the current PAD peer, the streaming request for these chunks is meaningless and the chunk's npf value needs to be decreased. As the npf notion indicates, the chunk's npf value will impact its replication priority at the downloading clients, so the PAD client needs to revise these chunks' npf values as soon as possible once they are not needed by the player. Then, the chunk with the smallest frequency value (the rarest) can be ranked higher in the chunk selection procedure. This rarest-first order chunk replication can guarantee the overall downloading performance of the eMule network.

2. For the chunks that have been played at the current PAD client, their replications have already been increased by one and their npf values also need to be reduced. With the increased availability of these chunks, their replication priorities should be definitely lowered for both streaming and downloading requests. This makes the chunk replication algorithm more efficient for both PAD and downloading clients.

As above described, in the WDCD algorithm two key parameters are used in the peer's chunk selection procedure. One is the frequency f and the other is the next-play-frequency npf. A larger f of the chunk indicates that there are more copies for this chunk in the eMule network, so the newly joining peer will get the chunk more easily and faster. These kinds of chunks will have the lower downloading priority. A smaller f means that the chunk is rare or very rare in the eMule network. It is hard for the other peers to download this chunk because there are not enough replications (sources) in the network, and the chunk will need to have a higher downloading priority. This

rarest-first policy will replicate all chunks of a file to as many peers as possible so that the newly joining peer can have as many sources of the file as possible. In turn, the receiving peer can download the file from different parallel sources to achieve the maximum downloading throughput. npf is different from *f*; the npf value indicates how urgently the PAD clients are waiting for the chunk to play it. A chunk with a larger npf value will be requested more urgently, so it must have a higher replication priority than the other chunks. Combining these two parameters, there are four possible cases to decide a chunk replication priority for the peer clients (Table 2).

Table 2 The chunk's replication priority assignment of the WDCD algorithm

Replication priority	npf	*f*
Priority 1	High	Low
Priority 2	Low	Low
Priority 3	High	High
Priority 4	Low	High

Priority 1 This is the worst chunk of the system and often happens at a flash crowd scenario. The chunk has a small number of sources or replications, but there are many PAD clients sending streaming requests for it. BASS discusses several proposals for this case. It is obvious that this kind of chunk should have the highest replication priority because it has the maximum number of waiting PAD clients but the rarest copies in the system. For either streaming requests or downloading requests, it should be replicated with the highest priority.

Priority 2 Although there are not too many waiting PAD clients for the chunk, the network still does not have enough copies at the peers. This chunk should have a lower replication priority than the chunk in the Priority 1 case. For this case, we should increase the chunk's availability by replicating it to more peers, which is the rarest-first replicating algorithm of the original eMule system. This rarest-first replication order benefits both the streaming and the downloading clients. With the increase of the chunk's availability, the later joining PAD clients can have the shorter startup time and fewer playback jitters. Certainly, the newly joining downloading clients can also obtain the faster downloading speed.

Priority 3 In this case, the chunk has a large npf value, meaning that there are many pending PAD clients streaming requests for it. But there are also enough replications of this chunk at the network because its frequency *f* is large. Thus, the third priority is given to this chunk's replication request although it is requested by the PAD clients very urgently. It is because the network has provided enough sources for this chunk.

Priority 4 It is similar to the case of Priority 3, but there are fewer PAD clients waiting for the chunk. Thus, we assign the lowest replication priority to it.

Based on the rules described in the above four cases, our WDCD algorithm proposes an enhanced chunk selection and replication procedure by combining the chunk's replication availability (defined by the *f* value) and the streaming request's density (defined by the npf value). This enhancement can add the streaming capability to the eMule network and has little impact on its downloading performance.

5 Experiment evaluation

Our algorithm is aimed to add streaming support to the eMule file sharing system, so we need to evaluate the performance for both streaming and downloading features. We use the startup time and playback jitters to evaluate the streaming performance. The startup time is defined as the time that one peer needs to wait to view a movie since the playback request is sent. A well-designed P2P VOD system must have a very short startup time to meet the user experience. In our experiments, the continuity index (CI) proposed in the CoolStreaming, defined as the number of pieces that arrive before the playback deadline compared to the total number of pieces, is used to measure the playback jitters. A larger CI means a better playback performance. Besides the streaming performance, the WDCD algorithm should still be able to keep the original eMule system's downloading throughput.

To verify the WDCD algorithm performance, we use the NTCUns (NCTUns, 2009) simulator to run and check the prototype implementation. The NCTUns is a Linux kernel based network simulator. It can run real applications directly from the simulated network node. It can provide the same P2P network

running environment as the real Internet based P2P framework. The prototype implementation is based on the aMule (aMule, 2009) open source project; we modify the aMule framework by adding our new algorithm. Because the NCTUns can run the aMule application as a normal Linux application from the simulated network node inside, it is very easy for us to design and implement our experiments. Since we can run the aMule applications directly in both the NCTUns simulator and the real Linux environment without any modification, we believe the experiment results achieved by NCTUns are the same as in the case of running over the real Internet.

In our experiments, the NCTUns is running on a Dell PC with Intel DuoCore and 2 GB DDR. Because NCTUns can support only a maximum of 64 network nodes at one PC, we set up an autonomous system (AS) network model with 64 nodes. One of them runs the aMule server, three are configured to run the aMule client as the seeder, and the remaining 60 nodes run the downloading clients or PAD clients. Every peer has a symmetrical network bandwidth and the upload and download bandwidths are both 512 kb/s. The seed file is a CBR movie encoded with a bit rate of 512 kb/s, and its total playback duration is 120 s. To make the experiment simpler and easier, we cut the file into pieces of chunk with 64 KB size, so every chunk has a playback duration of 1 s. Our model is evaluated in a synthetic scenario where all peers send the playback or downloading requests simultaneously. This is the classic flash crowd case in a VOD system. Every experiment lasts 200 s. The 60 peer clients send the requests at the same time for every experiment with different percentages of PAD clients. A total of 20 experiments were carried out.

In the evaluation experiments, the performance results of the eMule rarest-first algorithm are compared with those of the WDCD algorithm. In our experiments, we focus mainly on the results of Priority 3 and Priority 4 cases in Table 2 to analyze the streaming performance improvement and the impact of the downloading performance.

5.1 Startup time evaluation

We compare the startup time performances between the original eMule and the enhanced eMule with WDCD. The results (Fig. 4) indicate that the WDCD algorithm has significantly improved the peer's startup time; most of the peers can start the streaming within 30 s. Considering that the peer's initial buffering time is fixed as 10 s and all peers are requesting for the video stream from the seed nodes in flash crowd mode, this startup time performance is quite a large improvement. The original eMule implementation has a larger startup time delay because the chunks at the movie head do not have a higher priority than the chunks in other positions to arrive at the receiving peer's player buffer. In the original eMule system, a receiving peer obtains the chunks in the rarest first order, so there are chances by which the head chunks arrive at the player behind the tail chunks. Then the peer cannot start the playback until these head chunks are received. This chunk downloading order will greatly increase the peer's startup time. The worst cases are when the head chunks are received behind all the other chunks of the movie, and then the peer almost needs to wait for the playback until the full movie is downloaded completely. For the WDCD algorithm, the head chunks have higher priority to be replicated and can arrive at the streaming receiving peer before the playback deadline. At the beginning of the experiment, the PAD clients start to request the head chunks sequentially, and then the eMule server will increase these chunks' npf values to indicate these urgent streaming requests. When the seeding peers try to replicate the chunks to the whole eMule network, they will assign a higher replication priority to the chunk with a higher npf value. Thus, the movie

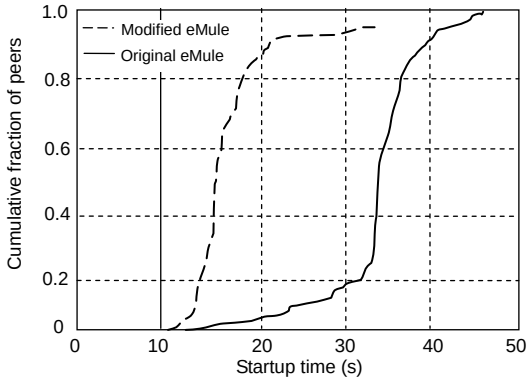

Fig. 4 Cumulative distribution of the startup time for the two different algorithms of all runs

The vertical line represents the earliest possible startup time when a peer could start the playback

head chunks will have some copies between the peers, and then the PAD clients have more chances to obtain the head chunks before the playback deadline in the enhanced eMule system and can have better startup time performance.

5.2 Playback jitter performance evaluation

Fig. 5 shows the CI as a function of PAD client's percentage. When the PAD clients' percentages are less than 60%, the streaming peers in the WDCD enhanced system have a larger CI and show a better playback performance. In the WDCD algorithm, the chunks with higher npf values have higher replication priorities, so the PAD client has more chances to obtain these chunks before the playback deadline. For the chunks with a zero npf value, their replication priority is decided by the frequency value. This is also helpful in smoothing the PAD client's playback because there are chances by which these chunks are downloaded at the next-play-window. When the PAD client number is increasing and the downloading client number is decreasing at the same time, it is harder for PAD clients to obtain the chunks before their playback deadlines. This is because we use the flash crowned model to simulate the peers' arrivals, and all the PAD streaming clients are requesting for the movie head chunks from the limited three seeders. At the beginning, only the head chunks are replicated between the peers because they have the highest priority, and there are limited sources for these chunks, so the PAD clients will face the content bottleneck at the player. This case is very much like the centralized client/server architecture. It will increase the initial startup time and the playback jitters.

We can conclude from the above results and analyses that the WDCD algorithm can achieve a good

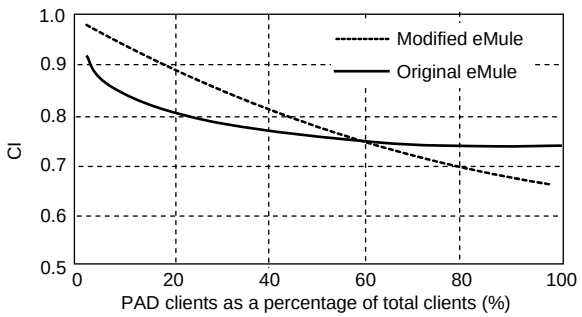

Fig. 5 Continuity index (CI) versus the PAD client number (all runs)

streaming performance when the PAD client percentage is lower than that of the downloading client. This scenario is the default behavior of the eMule system where most of the clients are the downloading ones who are just downloading the contents in the background without simultaneous streaming activities. Nowadays, more and more devices are available with the eMule downloading capability, such as the network attached storage (NAS) devices, which provide only the downloading function and do not support a streaming user interface. With the increase of these kinds of downloading only devices, the WDCD algorithm can enable the PAD clients of eMule to obtain better streaming preview experiences.

5.3 Downloading speed performance evaluation

Besides the streaming playback performance, we also need to evaluate whether the WDCD algorithm reduces the overall eMule downloading throughput. We compare the downloading speed of the original eMule implementation and the WDCD algorithm enhanced eMule system. The peer's downloading throughput value is very stable for the original eMule rarest-first algorithm as the percentage of PAD clients varies (Fig. 6). This is because the increasing streaming clients do not change the chunk's frequency value or replication priority. As for the WDCD algorithm, the peer's downloading speed decreases as the PAD client percentage increases. In our experiments the flash crowd mode is adopted, so all the peers are requesting for the chunks in playback timestamp order from the three seed nodes at the beginning. When the number of PAD clients is increasing, the head chunks of the file have higher replication priorities, and the tail chunks have little chance to be replicated. Thus, the peer must wait for a longer time to finish the downloading of the tail chunks. The overall downloading performance will not be recovered until all chunks' npf values reach zero after the 120 s playback duration. Finally the peers select and replicate the chunks in rarest-first order again.

The results in Fig. 6 support the same conclusion as drawn from Fig. 5 that the WDCD algorithm can have a better streaming playback performance while keeping a good overall downloading throughput when the PAD clients have a low percentage over the total clients.

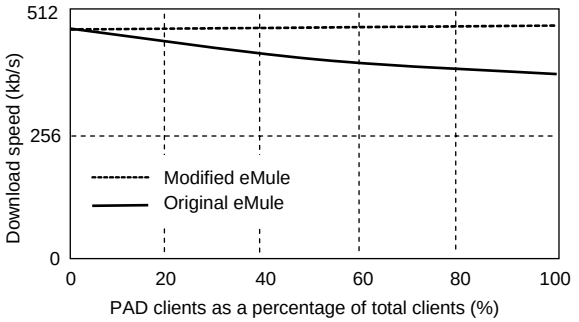

Fig. 6 Download speed versus the PAD client number (all runs)

6 Conclusions and future work

In this paper, we show a simple enhanced chunk selection and replication algorithm to add the playing-as-downloading capability to the eMule system. This algorithm is based on the original eMule rarest-first algorithm. It needs only a small modification and is easy to implement. We have deployed this modification at aMule source code and evaluated the prototype with the real application level simulators of NCTUns. The results show that the peer could achieve a good streaming playback performance and will not degrade the system's overall downloading throughput if the number of streaming clients is much smaller than that of the downloading clients.

In the future, we will modify the peer's arrival model in our experiment to learn more about the WDCD performance. The flash crowd arrival model is currently used for simplicity, but it is more often for the P2P system to use the Poisson distribution model to simulate the peer's arrival. An attempt will be made to compare the implementation and the architecture between the eMule and BitTorrent systems to study their suitability for the PAD feature.

Acknowledgements

The authors would like to thank Albert S. Wang (e-mail: albert_s_wang@yahoo.com), previously with Agilent Technologies and now a visiting fellow at the Advanced Digital Technology and Instrument Institute of Zhejiang University, for his contribution to this manuscript.

References

aMule, 2009. The aMule Open Source Project. Available from http://www.amule.org/ [Accessed on July 11, 2009].

Androutsellis-Theotokis, S., Spinellis, D., 2004. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, **36**(4):335-371. [doi:10.1145/1041680.1041681]

Annapureddy, S., Guha, S., Gkantsidis, C., Gunawardena, D., Rodriguez, P.R., 2007. Is High-Quality VOD Feasible Using P2P Swarming? Proc. 16th Int. Conf. on World Wide Web, p.903-912. [doi:10.1145/1242572.1242694]

Bickson, D., Dolev, D., Weiss, Y., 2007. Efficient Peer-to-Peer Content Distribution. Available from http://leibniz.cs.huji.ac.il/tr/858.pdf [Accessed on July 11, 2009].

Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A., 2003. SplitStream: High-Bandwidth Multicast in Cooperative Environments. Proc. 19th ACM Symp. on Operating Systems Principles, p.298-313. [doi:10.1145/945445.945474]

Cheng, B., Stein, L., Jin, H., Zhang, Z., 2008a. A Framework for Lazy Replication in P2P VoD. Proc. 18th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video, p.93-98. [doi:10.1145/1496046.1496068]

Cheng, B., Stein, L., Jin, H., Liao, X., Zhang, Z., 2008b. GridCast: improving peer sharing for P2P VoD. *ACM Trans. Multimedia Comput. Commun. Appl.*, **4**(4), No. 26. [doi:10.1145/1412196.1412199]

Cheng, B., Stein, L., Jin, H., Zhang, Z., 2008c. Towards Cinematic Internet Video-on-Demand. Proc. 3rd ACM SIGOPS/EuroSys European Conf. on Computer Systems, p.109-122. [doi:10.1145/1352592.1352605]

Chu, Y.H., Rao, S.G., Seshan, S., Zhang, H., 2002. A case for end system multicast. *IEEE J. Sel. Areas Commun.*, **20**(8):1456-1471. [doi:10.1109/JSAC.2002.803066]

Dana, C., Li, D., Harrison, D., Chuah, C., 2005. Bass: BitTorrent Assisted Streaming System for Video-on-Demand. Proc. IEEE 7th Workshop on Multimedia Signal Processing, p.1-4. [doi:10.1109/MMSP.2005.248586]

Do, T.T., Hua, K.A., Tantaoui, M.A., 2004. P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment. IEEE Int. Conf. on Communications, p.1467-1472. [doi:10.1109/ICC.2004.1312755]

Do, T.T., Hua, K.A., Tantaoui, M.A., 2008. Robust video-on-demand streaming in peer-to-peer environments. *Comput. Commun.*, **31**(3):506-519. [doi:10.1016/j.comcom.2007.08.024]

Füllemann, L., 2005. P2P Mechanism Design. Available from http://dcg.ethz.ch/theses/ss05/p2pmd_report.pdf [Accessed on July 11, 2009].

Ghose, D., Kim, H.J., 2000. Scheduling video streams in video-on-demand systems: a survey. *Multimedia Tools Appl.*, **11**(2):167-195. [doi:10.1023/A:1009681521536]

GnuTella, 2009. The Annotated Gnutella Protocol Specification v0.4. Available from http://rfc-gnutella.sourceforge.

net/developer/stable/index.html [Accessed on July 11, 2009].

Guo, Y., Suh, K., Kurose, J., Towsley, D., 2003. P2Cast: Peer to Peer Patching Scheme for VOD Services. Proc. 12th Int. Conf. on World Wide Web, p.301-309. [doi:10.1145/775152.775195]

Guo, Y., Suh, K., Kurose, J., Towsley, D., 2008. DirectStream: a directory-based peer-to-peer video streaming service. *Comput. Commun.*, **31**(3):520-536. [doi:10.1016/j.comcom.2007.08.022]

Huang, Y., Fu, T.Z.J., Chiu, D.M., Lui, J.C.S., Huang, C., 2008. Challenges, design and analysis of a large-scale P2P-VoD system. *ACM SIGCOMM Comput. Commun. Rev.*, **38**(4):375-388. [doi:10.1145/1402946.1403001]

Jannotti, J., Gifford, D.K., Johnson, K.L., Kaashoek, F.M., O'Toole, J.W.Jr., 2000. Overcast: Reliable Multicasting with an Overlay Network. Usenix Operating System Design & Implementation Symp., p.197-212.

Jiang, X.X., Dong, Y., Xu, D.Y., Bhargava, B., 2003. GnuStream: a P2P Media Streaming System Prototype. Proc. Int. Conf. on Multimedia and Expo, **2**:II-325-8. [doi:10.1109/ICME.2003.1221619]

Koo, S.G.M., Lee, C.S.G., Kannan, K., 2005. A Resource-Trading Mechanism for Efficient Distribution of Large-Volume Contents on Peer-to-Peer Networks. Proc. 14th Int. Conf. on Computer Communications and Networks, p.428-433. [doi:10.1109/ICCCN.2005.1523902]

Kostic, D., Rodriguez, A., Albrecht, J., Vahdat, A., 2003. Bullet: high bandwidth data dissemination using an overlay mesh. *ACM SIGOPS Oper. Syst. Rev.*, **37**(5):282-297. [doi:10.1145/1165389.945473]

Kulbak, Y., Bickson, D., 2005. The eMule Protocol Specification. Available from http://www.cs.huji.ac.il/labs/danss/p2p/resources/emule.pdf [Accessed on July 11, 2009].

Lee, G.J., Choi, C.K., Choi, C.Y., Choi, H.K., 2005. P2Proxy: Peer-to-Peer Proxy Caching Schema for VOD Services. Proc. 6th Int. Conf. on Computational Intelligence and Multimedia Applications, p.272-277. [doi:10.1109/ICCIMA.2005.42]

Leibowitz, N., Ripeanu, M., Wierzbicki, A., 2003. Deconstructing the Kazaa Network. Proc. 3rd IEEE Workshop on Internet Applications, p.112-120. [doi:10.1109/WIAPP.2003.1210295]

Liao, X., Jin, H., Liu, Y., Ni, L.M., 2007. Scalable live streaming service based on interoverlay optimization. *IEEE Trans. Parall. Distr. Syst.*, **18**(12):1663-1674. [doi:10.1109/TPDS.2007.70708]

Liu, J.C., Rao, S.G., Li, B., Zhang, H., 2008. Opportunities and challenges of peer-to-peer Internet video broadcast. *Proc. IEEE*, **96**(1):11-24. [doi:10.1109/JPROC.2007.909921]

Liu, Y., Guo, Y., Liang, C., 2008. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Network. Appl.*, **1**(1):18-28. [doi:10.1007/s12083-007-0006-y]

Magharei, N., Rejaie, R., 2007. PRIME: Peer-to-Peer Receiver-Driven Mesh-Based Streaming. INFOCOM 26th IEEE Int. Conf. on Computer Communication, p.1415-1423. [doi:10.1109/INFCOM.2007.167]

Magharei, N., Rejaie, R., Guo, Y., 2007. Mesh or Multiple-Tree: a Comparative Study of Live P2P Streaming Approaches. 26th IEEE Int. Conf. on Computer Communications, p.1424-1432. [doi:10.1109/INFCOM.2007.168]

Mol, J.J.D., Pouwelse, J.A., Meulpolder, M., Epema, D.H.J., Sips, H.J., 2008. Give-to-get: free-riding resilient video-on-demand in P2P systems. *SPIE*, **6818**:681804. [doi:10.1117/12.774909]

NCTUns, 2009. The NCTUns Network Simulator. Available from http://nsl.csie.nctu.edu.tw/nctuns.html [Accessed on July 11, 2009].

Pai, V., Kumar, K., Tamilmani, K., Sambamurthy, V., Mohr, A., 2005. Chainsaw: eliminating trees from overlay multicast. *LNCS*, **3640**:124-140. [doi:10.1007/11558989]

Pouwelse, J.A., Garbacki, P., Epema, D.H.J., Sips, H.J., 2005. The BitTorrent P2P file-sharing system: measurements and analysis. *LNCS*, **3640**:205-216.

Rodriguez, P., Tan, S., Gkantsidis, C., 2006. On the feasibility of commercial, legal P2P content distribution. *ACM SIGCOMM Comput. Commun. Rev.*, **36**(1):75-78. [doi:10.1145/1111322.1111339]

Sandberg, O., Wiley, B., 2000. Freenet: a Distributed Anonymous Information Storage and Retrieval System. Proc. Workshop on Design Issues in Anonymity and Unobservability, p.311-320.

Tian, Y.,Wu, D., Ng, K.W., 2008. A novel caching mechanism for peer-to-peer based media-on-demand streaming. *J. Syst. Archit.*, **54**(1-2):55-69. [doi:10.1016/j.sysarc.2007.03.008]

Tran, D.A., Hua, K.A., Do, T., 2003. ZIGZAG: an Efficient Peer-to-Peer Scheme for Media Streaming. 22nd Annual Joint Conf. of IEEE Computer and Communications Societies, **2**:1283-1292.

Vlavianos, A., Iliofotou, M., Faloutsos, M., 2006. BiTos: Enhancing BitTorrent for Supporting Streaming Applications. Proc. IEEE 25th Int. Conf. on Computer Communications, p.1-6. [doi:10.1109/INFOCOM.2006.43]

Zhang, X., Liu, J., Li, B., Yum, T.S.P., 2005. CoolStreaming/DONet: a Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming. Proc. 24th Annual Joint Conf. of IEEE Computer and Communications Societies, **3**:2102-2111. [doi:10.1109/INFCOM.2005.1498486]