2010 16th International Conference on Parallel and Distributed Systems

Multi-Task Downloading for P2P-VoD: An Empirical Perspective

Tieying Zhang*, Zhenhua Li†, Xueqi Cheng‡, Xianghui Sun§
*Institute of Computing Technology, Chinese Academy of Sciences
Email: zhangtiey@software.ict.ac.cn
†Department of Computer Science and Technology, Peking University
Email: lzh@net.pku.edu.cn
‡Institute of Computing Technology, Chinese Academy of Sciences
Email: cxq@ict.ac.cn
§China Telecom Group Corporation
Email: xianghui-sun@chinatelecom.com

***Abstract*—For current P2P-VoD systems, three fundamental problems exit in user experience: exceedingly large startup delay, long jump latency, and poor playback continuity. These problems primarily stem from lack of media data. In this paper, we propose Multi-Task Downloading with Bandwidth Control (MTD(BC)), an efficient and practical mechanism to prefetch media data. In MTD, a user can download multiple videos in parallel with its current viewing, which significantly decreases video switching delays. However, MTD brings a serious problem: downloading "other" tasks could impede the playback performance of the current viewing, especially in low-bandwidth network. This problem is solved through our design of bandwidth control. To our knowledge, we are the first to propose *MTD with bandwidth control* for P2P-VoD and conduct empirical evaluations in the real-world system. The running results show that MTD(BC) achieves better streaming quality than the traditional method. In particular, our mechanism reduces 75% of startup delay and 36% of jump latency in low-bandwidth network with high system scalability.**

***Index Terms*—Peer-to-Peer, Video-on-Demand, Data Schedule, User Experience, System Scalability**

I. Introduction

With the rapid deployment of broadband access into household, P2P video-on-demand (P2P-VoD) has become one of the most popular Internet applications. Several large-scale industrial P2P-VoD systems have been deployed over the Internet, such as PPLive [10], PPStream [6] and UUSee [7], etc. While hundreds of thousands of people began to use such video service, there still exist three fundamental performance problems: large startup delay, long jump latency, and poor playback continuity. Measurement studies of P2P-VoD systems [8], [10] have shown that the playback continuity has not been satisfactory enough, and the performance problems primarily stem from lack of media data. Therefore, how to efficiently acquire enough data is vitally important.

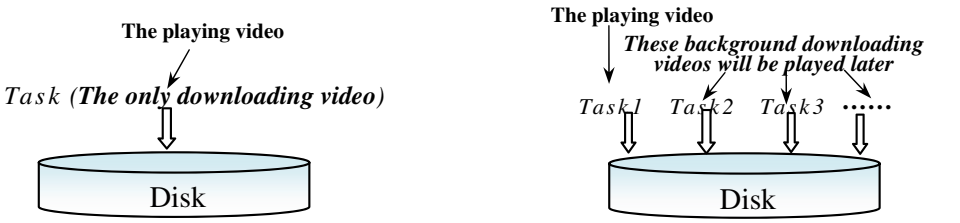
Currently, a common practice in P2P-VoD is to use Single-Task Downloading (STD) [6]–[8], [10]. In STD, as shown in Figure 1(a), only one video (task) is being downloaded at any time, where the downloading task is also the current viewing video. Intuitively, STD is a natural idea to download the viewing video. However, measurements [8], [10] have indicated that STD has long startup and jump delay which are typically on the order of 10 - 60 seconds. Particularly, a user always wants to watch some popular movies consecutively. For example, the top 3 popular movies are all the favorites for a user and he will watch them all. The switch delay of 10 - 60 seconds is certainly unacceptable, as users are used to the delays of less than 3 seconds when switching videos and channels [15].

Furthermore, lot of bandwidth is idle in STD. Only one downloading video usually can not make full use of peer download and upload bandwidth to share media data. Taking the home network ADSL for example (we refer to this kind of network as low-bandwidth network), the download bandwidth is usually 2 - 4 Mbit/s (Mbps) and the bit rate of a video is 350 Kbps - 500 Kbps for most P2P-VoD systems [6], [7], [10]. Therefore, only one video being downloaded can only take about 1/12 - 1/10 of download bandwidth. The "bandwidth waste" is more serious in high-bandwidth network, such as ADSL2+ and campus network. ADSL2+ is widely deployed throughout Europe and North America, and some other countries in the world [4]. It can be as high as 24 Mbps downstream and 3.5 Mbps upstream bandwidth [5]. As for the video bit rate of 500 Kbps, it is only about 1/50 of the download bandwidth and 1/7 of the upload bandwidth. That is to say, most of the bandwidth has been wasted in STD.

Triggered by the above observations, in this paper, we propose a cross-video downloading scheme, which we refer to as Multi-Task Downloading (MTD). MTD allows the user to download other videos (tasks) in parallel with his current viewing video, shown in Figure 1(b). The so-called "other" tasks are totally decided by the user himself, so they are also the candidate videos to be played sooner or later. When a peer switches from the current viewing to the background task, the startup content has been downloaded on the disk, so the new video can often be played at once. This advantage significantly decreasing the startup delays when switching videos.

Our mechanism can also improve the jump latency and playback continuity. Assume that if the video has been totally downloaded, any jump latency is 0 second. In our mechanism,

1521-9097/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPADS.2010.11



(a). Single-Task Downloading (STD) **(b). Multi-Task Downloading (MTD)**

Fig. 1. STD vs. MTD.

when a user starts to watch the background video, the video data has been totally or partially cached on the disk. When this user jumps to a new position in the video, the data is likely to be cached. Therefore, MTD can decrease the jump latency to some extent and the already prefetched data can also improve the playback continuity.

The idea of MTD is straightforward, but why such simple idea does not appear in the current P2P-VoD systems? The major concern about MTD is that *downloading multiple tasks could impede the playback performance of the current viewing video*, **especially in low-bandwidth network**. The problem stems from two reasons:

1) The download bandwidth of the peer is depletable. The parallel downloading in MTD could run out of download bandwidth and make the current viewing discontinuous. For a peer, its multiple tasks share its depletable download bandwidth. If there are a lot of tasks being downloaded in parallel, the download speed of the current viewing video can not be guaranteed. We call this phenomenon **local bandwidth competition**.

2) The total upload bandwidth of the suppliers is depletable. For the whole system, downloading multiple tasks could cause the aggregate requirements larger than aggregate supply. According to work [9], [11], in order to guarantee the QoS for the downloading streams, a P2P-VoD system should guarantee:

$$\sum_{i=1}^{n} d_i \leq u_s + \sum_{i=1}^{n} u_i \tag{1}$$

where d_i is the bit rate of the downstream of peer i, n is the number of peers exchanging media data, u_s is the server upload capacity, and u_i is peer i's upload capacity. If a peer can download multiple tasks in parallel, this could greatly increase $\sum_{i=1}^{n} d_i$ of the system, while $\sum_{i=1}^{n} u_i$ is changeless. When $\sum_{i=1}^{n} d_i$ is larger than $u_s + \sum_{i=1}^{n} u_i$ (u_s is a relative small constant), the QoS for the downloading streams can not be guaranteed. We call this phenomenon **global bandwidth imbalance**.

In this paper, we solve this problem through the design of *bandwidth control*. Our contributions can be summarized as follows:

1) We propose a simple and efficient cross-video downloading scheme to decrease startup delays, which we refer to as Multi-Task Downloading (MTD). Using MTD, besides the current viewing video, a user can also select some candidate videos to download in the background. Because of the already prefetched data, the candidate videos can be played immediately without delay.
2) Based on theoretical analysis, we design a *bandwidth control* mechanism to solve the problem involving in MTD. Through this mechanism, MTD is practical to be deployed in the real-world system.
3) We evaluate the performance of MTD via extensive experiments in our real-world system, CoolFish [1]. CoolFish has been deployed over 20 provinces in China. It has received over 4.8 million user visits in the last 20 months and the number of recent daily visits has exceeded 7000.

To our knowledge, we are the first to propose *MTD with bandwidth control* into P2P-VoD and implement it in the real-world system.

The rest of this paper is organized as follows. Section 2 describes our idea. Section 3 evaluates MTC's performances in CoolFish. Section 4 reviews the related work. Finally, in Section 5, we conclude this paper.

II. MTD WITH BANDWIDTH CONTROL

A. Basic Idea Based on Observations

Actually, we implement STD mechanism in the first version of CoolFish, just like most commercial P2P-VoD systems. Then we conduct a basic measurement of bandwidth utilization of the system. In CoolFish, 80% users are from CSTNet as shown in Table 1. Our measurement mainly focuses on these users. They have the download and upload capacity of 24Mbps and 3.5Mbps respectively [2], the same as ADSL2+. We record the average download and upload speed every 1 minute. Figure 2 represent the bandwidth utilization over 24 hours. We can see that the bandwidth utilization of both download and upload are very low. Even in hot time (from 8:00 to 22:00), STD can only utilizes 15% download bandwidth and 20% upload bandwidth in average. This phenomenon inspires us to design a more reasonable mechanism to make full advantage of peer's bandwidth.

Multi-Task Downloading is a natural idea to utilize the free bandwidth. In MTD, besides the current viewing video, a user can also select candidate videos downloading in the

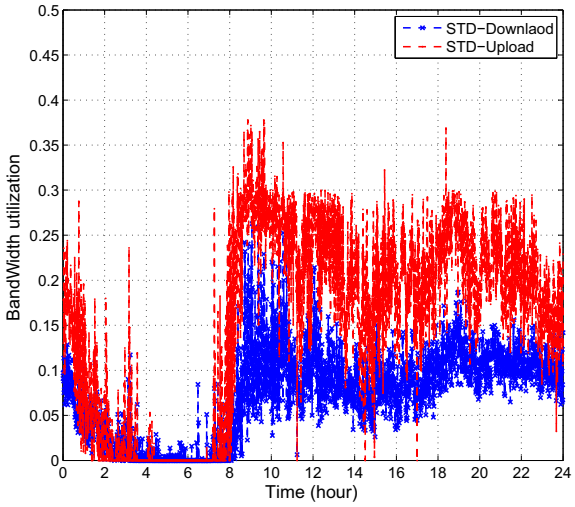


Fig. 2. Bandwidth utilization in STD.

background. As shown in Figure 1(b), task1 is the current playing video, and the other tasks being downloaded in the background are selected by the user.

Some people might ask what videos are selected for background download. The most favorable way is to predict which video a user will pick next. However, such prediction is difficult when there are many channels (videos) as verified in work [13]. Instead, in our method the background tasks are totally decided by user himself. It is imaginable that these background tasks are selected according to the user's interest. Thus, these videos will be played sooner or later. When the user switches video, because we already prefetched the data, it can be played immediately without delay.

B. Bandwidth Control

Our basic idea is simple and straightforward. But why such simple idea does not appear in the current P2P-VoD systems? The major concern about MTD is that downloading "multi" tasks could impede the playback performance of the current viewing, especially in low-bandwidth network. In this section, we design bandwidth control to guarantee the playback continuity of the current viewing. Intuitively, we should limit the download speed of the background tasks. But how to get the upper bound of the "limit"? Our main task in bandwidth control is to determine the upper bound of the background download speed. To this end, we introduce the following notations:

n: the total number of peers;
p_i: the i-th peer (i=1, 2, ..., n);
r_i: the bit rate of p_i's current viewing;
d_i: the download capacity of p_i;
u_i: the upload capacity of p_i;
u_s: the upload bandwidth of the server;
There exit two kind of downloading tasks: the current viewing task and the background tasks. Therefore, we divide the total downloading speed of into:

d_i^{View}: the download speed of p_i's current viewing;
d_i^{Down}: the download speed of p_i's background tasks;
UBd_i^{Down}: the upper bound of d_i^{Down};

As explained in Section 1, two reasons should be considered to solve the problem: **local bandwidth competition** and **global bandwidth imbalance**. In this section, we design bandwidth control for these two aspects.

1) Bandwidth Control for "local bandwidth competition": For local bandwidth competition, it is easy to get the upper bound of the "limit". In order to guarantee the bit rate of p_i's current viewing, the upper bound of the background download speed is :

$$\text{UB}d_i^{Down} = d_i - r_i \tag{2}$$

2) Bandwidth Control for "global bandwidth imbalance": Usually the default value of is a relative small value for a P2P system. In this case, we define as a constant number c. From the Formula (1), we have:

$$\sum_{i=1}^{n} d_i \leq c + \sum_{i=1}^{n} u_i \tag{3}$$

As mentioned above, every peer has two kind of d_i: d_i^{View} and d_i^{Down}. Then Formula (2) is equal to:

$$\sum_{i=1}^{n} d_i^{View} + \sum_{i=1}^{n} d_i^{Down} \leq c + \sum_{i=1}^{n} u_i \tag{4}$$

Subject to:

$$d_i^{View} \geq r_i \tag{5}$$

From Formula (3) and (4) we have:

$$\sum_{i=1}^{n} d_i^{Down} \leq c + \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} d_i^{View} \leq c + \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} r_i \tag{6}$$

From Formula (5), we get the upper bound for all peers' background speed:

$$\text{UB}\sum_{i=1}^{n} d_i^{Down} = c + \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} r_i \tag{7}$$

Next, we need to determine the upper bound value for each peer.

Because every peer has a different upload capacity, the download speed should also have the corresponding ability in order to share the media data. Therefore, from Formula (6) we can determine the upper bound of as:

$$\text{UB}d_i^{down} = \frac{(c + \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} r_i) \cdot u_i}{\sum_{i=1}^{n} u_i} \tag{8}$$

where $\frac{u_i}{\sum_{i=1}^{n} u_i}$ is the share ability proportion of p_i.

3) Bandwidth Control in Practical System: To sum up, we get the upper bound of background download speed for each peer:

$$\mathrm{UB}d_i^{down} = \mathrm{Min}\{d_i - r_i, \frac{(c + \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} r_i) \cdot u_i}{\sum_{i=1}^{n} u_i}\} \quad (9)$$

The value of $\mathrm{UB}d_i^{down}$ is computed by a tracker, and is updated to the peers every once in a while. One immediate concern for the bandwidth control design is its update overhead. Actually, after $\mathrm{UB}d_i^{down}$ is computed by the tracker, it is not updated to peers by broadcast. Instead, this value is capsulated in the normal control messages, and such kind of message only transmits every once in a while. For example, in our CoolFish system, the interactive messages transmit every 2 minutes. Therefore, the overhead brought by the bandwidth control is very low. This will be proved by the experiments.

In the following sections, we call MTD with bandwidth control as MTD(BC), and the simple MTD without bandwidth control as MTD(no-BC).

III. Performance Evaluation

The evaluation of MTD is based on our real-world P2P-VoD system, CoolFish. In this section, we first give an overview of the CoolFish system. Then, the metrics and configuration are presented. Finally, we discuss the result in detail.

A. System Overview

CoolFish is mainly deployed in China Science & Technology Network (CSTNet), a nationwide network connecting about 200 research institutes of the Chinese Academy of Sciences and four campuses with more than 58,000 students. It has been very popular since released in CSTNet for most communication traffic locates in the same ISP according to our observation. From Oct. 2008 to Mar. 2010, there have been over 4.2 million user visits and the number of recent daily visits has exceeded 7000. Its users mainly come from 4 ISPs in China: CSTNet, CERNet, Telecom and Netcom. Our measurements show that 80% users come from CSTNet and CERNet. The remaining 20% users belong to Telecom and Netcom. At hot time, there are over 700 simultaneous viewers. CoolFish is able to support an average video bit rate of 700Kbps, which is about 50% higher than that of most commercial P2P-VoD systems with a video bit rate less than 450Kbps [6], [7]. Table I presents the detailed log statistics of the system.

CoolFish is a mesh-based network just like BitTorrent system. The implementation of CoolFish is based on our previous P2P VoD system [17] which only utilized the STD mechanism. Then we tried MTD(no-BC) in the second version, and MTC(BC) in the third one with over 80,000 lines of C++ codes in total. To guarantee the QoS, a media content server is deployed in CoolFish to provide more than 1,500 videos. A peer will require data from content server if it can not get the data from other peers. When a peer joins the system, it can also share its local videos. To simplify the cache system, we do not restrict the cache capacity in CoolFish. Which video should be deleted is totally decided by user himself. There is another server in the system, Tracker, which is responsible for the control messages reported from peers every two minutes. The peers are distributed over 20 provinces in China, shown in Figure 3 from Google Analytics report [3].

TABLE I
CoolFish system statistics from Oct. 2008 to Jun. 2010.

Parameter	Value
Total number of visited users	≈4,800,000
Peak number of online users	> 700
Server upload bandwidth	100 Mbps
Number of videos	> 1000
Average video bit rate	700 Kbps
Average video length	1.2 hours
Average disk space contribution per peer	3 GB
Percentage of CSTNet and CERNet users	80%
Percentage of NAT users	22%

Fig. 3. CoolFish user distribution in China (from google analytics).

B. Metrics and Configurations

We evaluate the performance of MTD from two aspects: *user experience* and *system scalability*. User experience includes three metrics: 1) playback continuity, 2) startup delay and 3) jump latency. System scalability mainly refers to 4) server stress and 5) control overhead. P2P-VoD systems pursue to achieve high system scalability with a guarantee of user experience.

1) *Playback continuity*: the ratio of pieces that arrive before or on playback deadlines.
2) *Startup delay*: the time from the moment a user sends a request for a video to the moment it starts playing the required video, after buffering 20-second data (20-second is an empirical value in CoolFish).
3) *Jump latency*: the time from the moment a user launches a jump operation to the moment it starts playing the video from the jump position after buffering 20-second data.
4) *Server stress*: the upload speed required at the media server to support the whole system. We use the peak stress to examine the system scalability.

5) *Control Overhead*: all control messages between peers and tracker, including the messages of joining, startup a video, jump to new position, data scheduling (and bandwidth control messages for MTD(BC)).

Up to now, there have been three versions of downloading scheme on CoolFish: 1) STD, 2) MTD without bandwidth control (MTD(no-BC)), and 3) MTD with bandwidth control (MTD(BC)). STD was used in the first implementation of CoolFish. We separate MTD(no-BC) and MTD(BC) in the second and third version of CoolFish in order to study the performance of bandwidth control.

In order to evaluate the three downloading schemes, we collect two-month log data for each scheme separately. From Figure 4, we can see that the system scales are similar for the three versions in these time intervals (we only represent the system scale in one week for clarity). In order to further guarantee the fairness of comparison, we did not add any other functions or improvements into the system.

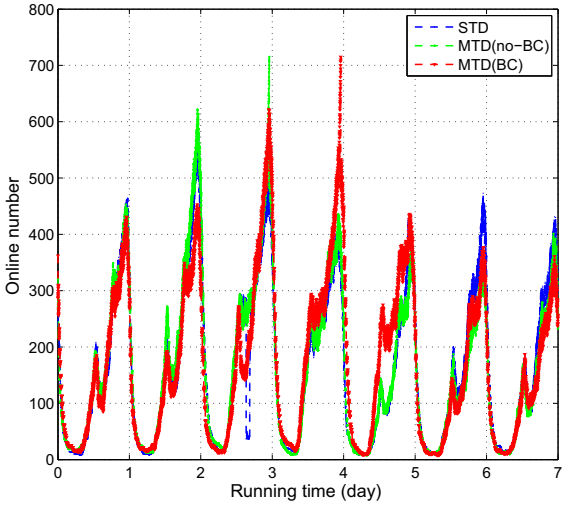


Fig. 4. Online number of peers over running time.

In CoolFish, the 80% users are in CSTNet, whose bandwidth ability is similar to ADSL2+ [2]. That is to say, there users have 3.5Mbps upload bandwidth. We call these users *high-bandwidth* users. On the other hand, the rest 20% users are in ASDL, we call these users *low-bandwidth* users. Because the bandwidth between these two kinds of users differs greatly, in order to be more accurate, we conduct the evaluations for them separately. How to know the upload bandwidth seems difficult for individual, but video providers can obtain it from the local ISPs. In our system, the upload bandwidth is from the "topology information table", which is maintained by China Telecom. Every item in the table is a pair of {ip, upload_bandwidth}. For the peers behind the same NAT, we consider each peer's upload bandwidth as the average value, by dividing the NAT bandwidth equally.

C. Evaluation results

1) Playback continuity: We track the playback continuity with different number of online peers. For high-bandwidth users (Figure 5), all the three curves show high playback continuity between 0.9 - 1.0. Because the network bandwidth is sufficient, the required data can be obtained in time. In contrast, for low-bandwidth users (Figure 6), the results differ greatly. STD and MTD(BC) still has high playback continuity of more than 0.9, but MTD(no-BC)'s continuity is extremely low (only around 0.7). The reason is 1) the parallel downloading in MTD(no-BC) runs out of bandwidth and makes the current viewing video discontinuous; 2) without bandwidth control, MTD could lead to the aggregate requirements larger than aggregate supply. On the other hand, MTD(BC) performs event better than STD.

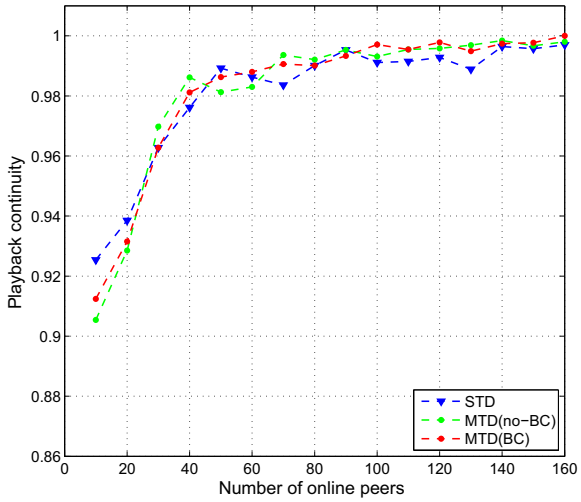


Fig. 5. Playback continuity for high-bandwidth users.

2) Startup delay: Figure 7 shows the PDFs of startup delay for high-bandwidth users. We observe that, MTDs have a big advance over STD. For both MTD(no-BC) and MTD(BC), the majority of startup delays are around 0 to 1 second (77% for MTD(no-BC), 84% for MTD(BC)). That means using MTDs, users can startup the video nearly at once. The reason is that MTDs can prefetch multiple videos and when users launch these already prepared videos, the startup delay is certainly small. However, for STD the startup delays of 0 to 1 second only hold 4%. The statistics in Table II describe the results in detail.

For low-bandwidth users (Figure 8), MTD(BC) still has low startup delays (0s - 1s hold 72%). However, MTD(no-BC) performs not as well as in high-bandwidth. This is because multi-task without bandwidth control could affect the current viewing, but under high-bandwidth this problem can be alleviated to trivial. That is why the results are so different.

3) Jump latency: For high-bandwidth (Figure 9), more than 50% jumps only need a latency of 0s to 1s for both MTD(no-

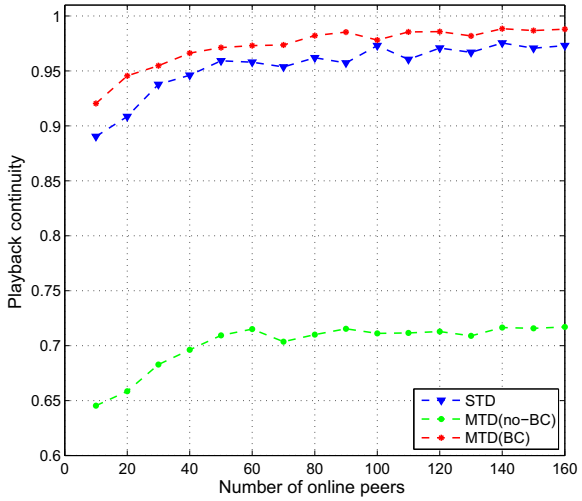


Fig. 6. Playback continuity for low-bandwidth users.

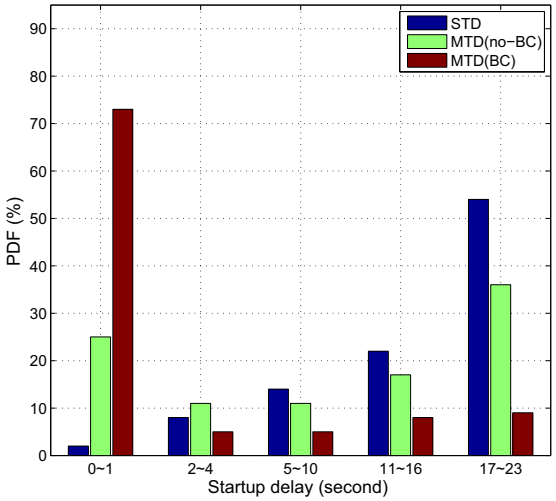


Fig. 8. Startup delay for low-bandwidth users.

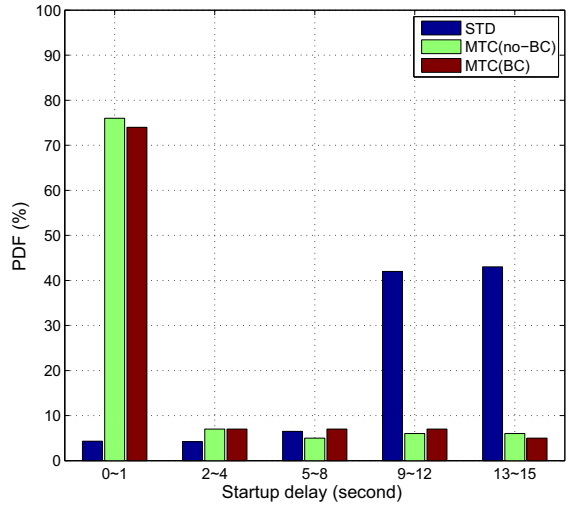


Fig. 7. Startup delay for high-bandwidth users.

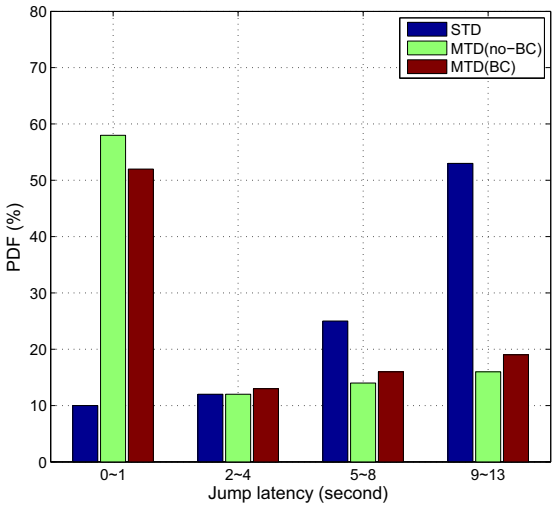


Fig. 9. Jump latency for high-bandwidth users.

BC) and MTD(BC). largely decreasing the jump latency than STC. The reason is MTD downloads not only the startup data but also the left data of the video.

For low-bandwidth users, different results are illustrated in Figure 10. MTD(no-BC) performs the worst in the three schemes, only about 2% jumps fall into 0s - 1s. Again this is because the parallel downstreams make the bandwidth insufficient under low-bandwidth environment. However, MTD(BC) still have 24% jumps falling into 0s - 1s.

4) Sever stress: There is a content server with 100Mbps upload bandwidth. If peer has not received required data when time out occurs, it will ask server to send data. Figure 11 shows the results. MTD(no-BC) performs the worst. The reason is the higher delays of startup and jump and poor playback continuity, leading to the urgent requests time out. But an interesting observation is that MTD(BC) achieves lower server stress than STD. Intuitively, multiple downstreams should bring more requests to server, leading high server stress. However, for MTD(BC), besides the bandwidth control, it provides more resources. Therefore, a peer has more candidates to select.

5) Control overhead: Besides the content server, there is another server in the system, Tracker. Tracker is responsible for the control message with all peers. In order to evaluate the update overhead of MTD(BC), we record all the messages between peers and Tracker. As shown in Figure 12, the control over head of the three schemes all increases linearly with the

TABLE II
AVERAGE VALUES OF EVALUATION RESULTS OF STD, MTD(NO-BC) AND MTD(BC).

	High-bandwidth users			Low-bandwidth users		
Metrics	**STD**	**MTD(no-BC)**	**MTD(BC)**	**STD**	**MTD(no-BC)**	**MTD(BC)**
Avg startup delay	11.7s	2.7s	2.8s	14.4s	11.2s	3.6s
Avg jump latency	7.8s	3.3s	3.8s	9.8s	11.9s	6.3s
Avg playback continuity	0.98	0.98	0.98	0.95	0.7	0.97

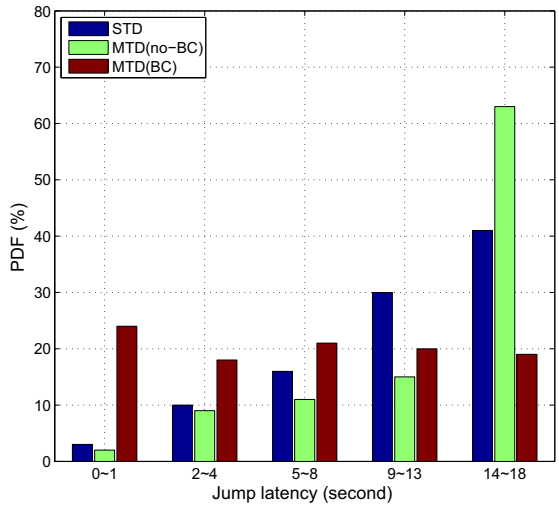


Fig. 10. Jump latency for low-bandwidth users.

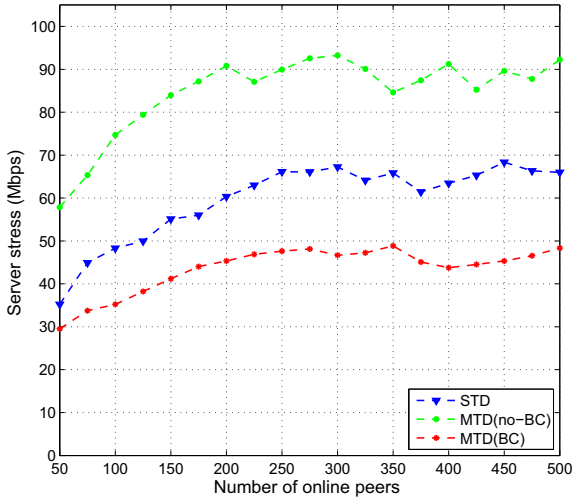


Fig. 11. Server stress.

number of peers, and there is no big difference among them. This is because the update messages are not sent by broadcast. Instead, they are capsulated in the normal control messages, and only transmits every once in a while. In our CoolFish system, the interval is 2 minutes. Therefore, the overhead brought by the bandwidth control is low.

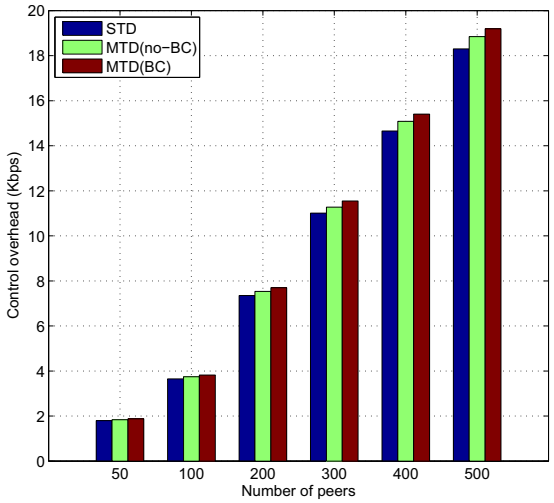


Fig. 12. Control overhead.

D. Analysis

As can be seen from the measurements, in high-bandwidth environment there is no large difference between MTD(no-BC) and MTD(BC), and they both performs largely better than STD in startup delays and jump latency. But the results are different in low-bandwidth environment. MTD(no-BC) performs the worst while MTD(BC) is still the best with the help of bandwidth control.

In practice, because a large part of normal household users are in low-bandwidth, MTD(no-BC) is not practical for the commercial P2P-VoD systems. On the contrary, MTD(BC) has good performance in both high-bandwidth and low-bandwidth. Thus MTD(BC) is a real practical mechanism.

Besides, one may argue the cache scheme in our CoolFish system. For the current P2P-VoD systems, they usually contribute a part of peer's disk space (usually 1GB) [6], [7], [10] to cache the content it has downloaded. However, how to arrange such 1GB storage is a challenging topic, and there has been no effective method yet. Instead, we do not restrict the size of cache in CoolFish system. The video which should be deleted is totally decided by user himself. This strategy not only simplifies the system design but also contributes abundant user storage. We can see from Table 1 that in CoolFish every user contributes 3GB storage, while other existing P2P-VoD systems [6], [7], [10] only contribute 1GB storage.

IV. Related Work

P2P download and cache scheme has attracted lots of research efforts. In [14], Wu et al. designed an algorithm to adjust the supply of server bandwidth to different channels. In [12], Gan et al. proposed an incentive mechanism to stimulate peers within spare bandwidth in resource-rich channels to help peers within resource-poor channels. However, the above works address the problems in P2P-live streaming. In live streaming, the bit rate for a given channel is constant which is different from that in VoD. Therefore, the methods of bandwidth allocation proposed in [14] and [12] are not suitable for P2P-VoD. VMesh [16] proposed a segment cache mechanism for P2P-VoD based on video popularity. However, the additional cache content is not selected by user it-self and can not improve the startup delays.

The closest works to ours are [8], [10] and [15]. The study in [8], [10] highlights the importance of caching and compares two cache mechanisms named SVC (single-video cache) and MVC (multi-video cache). Nevertheless, they are totally different from STD and MTD. SVC and MVC focus on cache mechanisms while ours aim at downloading schemes. SVC means a peer only caches the current viewing video on his disk, while MVC can also cache multiple videos which have been watched before. Such function of MVC is already included in our schemes. As mentioned in Section 3.4, we cache all the videos in the disk. However, what we are addressing are not cache mechanisms but downloading schemes. In [8], [10], the downloading task of MVC is still the current viewing video, which is largely different from ours. And the comparison results of SVC and MVC are based on trace-driven simulations while ours is conducted using a real deployed P2P-VoD system. [15] presents a live streaming design: VUD (view-upload decoupling) and conducts comparisons with ISO (isolated-channel). VUD focuses on bringing stability to multi-channel systems. It strictly decouples peer downloading from uploading. That means a peer will receive some content it do not need but uploads to other peers. That differs from ours. In our design, the content downloaded is what the peer will watch later. Further more, VUD is designed for P2P live streaming rather than VoD. In a P2P live streaming system, the users viewing the same channel are synchronous and cannot jump in the channel, while P2P-VoD allows asynchronization and jump operations.

V. Conclusion

In this paper, we present MTD with bandwidth control (MTD(BC)), an efficient and practical downloading scheme for P2P-VoD system. By prefetching data of multiple videos, a majority of startup delays can be decreased to 0s - 1s. With the considerate design of bandwidth control, MTD is practical in both high-bandwidth and low-bandwidth network. Using a real-world P2P-VoD system, we conduct an in-depth analysis of STD, MTD(no-BC) and MTD(BC). We find that compared with the traditional method of STD, MTDs have a distinct advantage in high-bandwidth networks. However, MTD(no-BC) underperforms STD in low-bandwidth network. In contrast, compared with STD, MTD(BC) can reduce more than 75% of startup delay on average, and 36% of jump latency in low-bandwidth network. Furthermore, MTD(BC) can achieve less sever stress with low control overhead. We believe that the results and analysis are helpful to the further P2P-VoD study.

Acknowledgment

This research is supported by the National Basic Research Program of China (Grant No.2011CB302305) and the National Natural Science Foundation of China (Grant No. 60873051 and No. 60933005).

References

[1] CoolFish web site. *http://www.cool-fish.org*.
[2] CSTNet billing rules. *http://www.cstnet.net.cn/bill.jsp*.
[3] Google Analytics. *http://www.google.com/analytics*.
[4] http://en.wikipedia.org/wiki/itu_g.992.5.
[5] http://www.itu.int/rec/t-rec-g.992.5/en.
[6] PPStream web site. *http://www.ppstream.com*.
[7] UUSee web site. *http://www.uusee.com*.
[8] B. Cheng, X. Liu, Z. Zhang, H. Jin, L. Stein, and X. Liao. Evaluation and optimization of a peer-to-peer video-on-demand system. *Journal of Systems Architecture*, 54(7):651–663, 2008.
[9] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay muilticast architecture. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, page 67. ACM, 2001.
[10] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p-vod system. *ACM SIGCOMM Computer Communication Review*, 38(4):375–388, 2008.
[11] R. Kumar, Y. Liu, and K. Ross. Stochastic fluid theory for P2P streaming systems. In *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 919–927, 2007.
[12] G. Tan and S. Jarvis. Inter-overlay cooperation in high-bandwidth overlay multicast. In *Parallel Processing, 2006. ICPP 2006. International Conference on*, pages 417–424, 2006.
[13] J. Wang, J. Pouwelse, J. Fokker, and M. J. T. Reinders. Personalization on a peer-to-peer television system. *Multimedia Tools and Applications*, pages 89–113, 2007.
[14] C. Wu, B. Li, and S. Zhao. Multi-channel live P2P streaming: Refocusing on servers. In *Proceedings of IEEE INFOCOM*, 2008.
[15] D. Wu, Y. Liu, and K. Ross. Queuing network models for multi-channel p2p live streaming systems. In *Proceedings of IEEE INFOCOM*, 2009.
[16] W. Yiu, X. Jin, and S. Chan. VMesh: Distributed segment storage for peer-to-peer interactive video streaming. *IEEE journal on selected areas in communications*, 25(9):1717–1731, 2007.
[17] T. Zhang, J. Lv, and X. Cheng. Mediacoop: Hierarchical lookup for p2p-vod services. In *ICPP '09: Proceedings of the 2009 International Conference on Parallel Processing*, pages 486–493, Washington, DC, USA, 2009. IEEE Computer Society.