2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Media streaming in P2P networks based on BitTorrent

Ankit Charls [1], Tushar Sharma[2] and P.K.Singh[3]

Atal Bihari Vajpayee- Indian Institute of Information Technology and Management, Gwalior
[1]ankitcharles@gmail.com,[2]tushar.sharma.9@gmail.com,[3]pksingh@iiitm.ac.in

Abstract. BitTorrent in the recent years has been one of the most effective mechanisms for the P2P content distribution. Although BT was created for the distribution of time insensitive content, in this paper we try to find out what are the changes that are required in-order to incorporate streaming in it. The importance of this capability is that peer will now have ability to start enjoying video before the complete download of the file. We propose peer to peer multimedia streaming solution based on the Bittorrent content distribution protocol. Bittorrent uses rarest block download first policy [1] which is not favorable for media streaming. Our approach gives higher priority to the blocks that are at the earlier point playback point. This is in contrast with the Bittorrent protocol which uses rarest block download first. Achieving this goal requires: (1) a piece selection strategy that effectively mediates the conflict between the goals of high piece diversity, and the in order requirements of media file playback, and (2) a new incentive mechanism, which supports media streaming**.**

Keywords: Bittorrent, rarest block download first, tit-for tat incentive mechanism, blocks, media streaming.

1. Introduction

Scalable on-demand streaming of stored media can be achieved using scalable server protocols such as patching and Hierarchical Stream Merging, server replication as with content distribution networks (CDNs), and/or peer-to peer techniques. This paper concerns peer-to-peer approaches. BitTorrent is a P2P file sharing protocol developed by Bram Cohen [1]. During the last few years Bittorrent has been proved to be a very successful content distribution P2P protocol. The success of BT lies on its ability to distribute content quickly by utilizing the capability of all the peers in peer to peer BT network. This ability comes from the mechanisms that provide incentive to the peers to contribute to the BT community preventing them from becoming free riders. Can BitTorrent be modified to support streaming? This is the question we address here. In our work we enhance BT with a view as you download**.** We want a peer to start reproducing the video content that is currently downloading, before it's completely get downloaded. This is very beneficial for the peer, because-

1. It reduces the time needed require to start enjoying the file.
2. It allows the peer to evaluate the quality of the video

P2P solutions can enable efficient and scalable media streaming, as long as they can meet the requirement of the sequential playback demand of the media streaming applications, which differs from file downloading, for which P2P network was originally designed. Recently P2P network has been used successfully for the live media streaming, but P2P paradigm is also applicable for more difficult case of on demand media streaming, which has received little attention. The two scenario share several common challenges, including the sequential playback demand of the large media objects, the geographic diversity. On demand streaming of the stored media files differs from the live media streaming in subtle way. First live streaming involves typically single streaming source, whereas stored media object can involve several streaming sources. Second live streaming allows peers to join at any time, without receiving the other portion of stream. The stored media case involves receiving the entire media object. Live streaming implicitly involves sustain content delivery at the intrinsic media playback rate*,* while the stored media case is general: the retrieval could vary (e.g. Slower than, faster than, or the same as media playback rate). These characteristic can challenge the performance of any existing P2P protocol. For example BitTorrent improves the efficiency of the file downloads by using the" Rarest Piece Download First" to improve the diversity of the pieces available in the network. However, streaming protocol require in order playback of the media content, which

naturally implies that in order retrieval of the pieces is desirable (but not strictly required). In-order collection of the pieces may reduce the spatial and temporal diversity of the pieces in P2P network, resulting in poor system performance. Research questions motivating our work are:
P1. Can BitTorrent like protocols provide efficient and scalable on-demand media streaming.
P2. What is the user perceived performance (i.e. start up delay, playback delay) in such a P2P networks?
The main contributions of this paper are as follows:
1. We present a highest priority download block first to replace rarest block download first. This approach solves the problem P1 associated with original BT.
2. We also propose a modified tit- for- tat incentive mechanism to improve streaming speed.

1.2 BITTORRENT

BitTorrent's goal is to distribute fast and efficient large files by using the upload bandwidth of the downloading peers. BT is using swarming techniques, in which the torrent file (the content that is distributed), is split in pieces (typically 256KB in size). In that way, peers can simultaneously download pieces from other peers. While the peer is downloading pieces of the file, it uploads the pieces that it has already acquired to its peers. Each time the peer has new piece, it advertises this information to its peer set (the peers that the peer is connect to). The only centralized component of BT is an entity called tracker. The tracker is responsible to help the peers find each other and to keep the download/upload statistics of each peer. Moreover peers during their initialization they retrieve from the tracker information about the file, such as the number of pieces that the file is split, the hashes of each piece (for integrity verification), etc. The strength of BT lies in its ability to resist to the Free-Riders phenomenon, in which selfish peers choose only to download the file without uploading. BT uses a Tit-for-Tat policy, where each peer chooses to upload to its peer as long as it takes something in return. If the neighbor peer behaves selfishly the Choking mechanism is invoked and the peer stops uploading to its neighboring peer. BT distinguishes peers into two categories, the seeds and the leechers. Seeds are peers that have already the whole file and leechers are peers that are in the progress of downloading the file. As soon as a leecher has downloaded the whole file, it becomes a seed. Another vital mechanism of BT is the Piece Selection mechanism. Peers always select to download the rarest Pieces within their peer set. This provides fast replication of the rarest pieces and ensures that the torrent file won't become easily extinct, in case a peer that has these particular pieces leaves.

2. Related Work

Prior work in the peer to peer streaming can be classified into either live streaming or on-demand streaming. These system typically use a tree based system or data driven approach. Tree based approach are typically based on application level multicast architectures, in which data is propagated through one or more relatively static spanning trees. Such application level solutions have mainly been used for live streaming. Related tree based approaches using cache and relay has also been proposed for on demand streaming. In cache and relay Systems, each peer receives content from one or more parents, and store it in local cache, from which it can later be forwarded to the clients that are at an earlier playback point of the file. The tree base approaches works best when peer connections are relatively stable. In data driven approach, distribution path are dynamically determined based on data availability. By splitting the file into smaller parts, each of which may take completely different path, data driven protocols can function effectively in dynamic environments (e.g., where peer may leave or join the system frequently, and peer connections are heterogeneous, with highly time varying bandwidths). With most peers at the similar playback points, peers in live streaming can typically exchange pieces effectively using a relatively small window of pieces. In contrast, with on demand streaming systems, peers may be at very different playback points. While *download* systems benefits from high piece diversity (as achieved rarest first policy), in the streaming pieces in sequential order. To achieve a compromise between these two objects, Annapureddy[2] propose splitting files into sub files, with each encoded using distributed network coding, and download using a BitTorrent like approach. By downloading sub files sequentially, playback can begin after the sub files have been retrieved. Use of large sub-files results in large startup delays, while using very small sub-files results in close to sequential piece retrieval, which can lead to poor performance. The best choice of sub-file sizes would be workload (and possibly also Client) dependent, although the method requires these sizes to be statically determined. The authors do not elaborate on how the sizes can be chosen, or how startup delays can be dynamically determined. Rather than statically splitting each file into sequentially retrieved sub files, Carlsson and Eager [3] propose a probabilistic piece selection policy with bias toearlier pieces. In another work Carlsson, Mohanti and Williamson [4] have analyzed the bittorrent protocol for on demand media streaming. They have created a model on the basis of Qiu and Srikant's [5] model. In this model they showed that number of downloader and seeds in the system are linearly dependent on the peer arrival rate. They have also compared rarest first and in-ordered download schemes. Hafeeda and Bhargava [6] proposed another approach formedia streaming which is very similar to BiTtorrent scheme. An attempt [7] tries to provide streaming service by using a hybrid server/P2P streaming system approach. The clients retrieve the stream from a dedicated server which in parallel share pieces using BT. BT protocol remains almost unaltered with the only modification that clients won't download any data prior to the current playback time. This work differs from ours, due to existence of dedicated streaming server. In our approach we consider only BT as the primary mechanism for Streaming. Moreover in [7] they have stated that BT is not suitable for streaming .The reason is that peers will have only sequential pieces of the stream and thus Tit-for-Tat will fail. Another interesting work is Cool Streaming [8]. Cool Streaming, uses a data centric design of an overlay network. The decision of requesting a particular piece is made based on a heuristic scheduling algorithm, which is similar to the Piece Selection mechanism of BitTorrent. Other interesting work is chainshaw [9], which uses BT concept but also uses gossip and pushed-based approaches that deviate from BT mechanisms.

2.1BitTorrent's limitations in Streaming

In this section we identify the limitation of the BitTorrent in providing streaming services and describe how streaming can be possible in BitTorrent. Earlier we have stated that BitTorrent uses rarest block download first piece selection policy. Although this policy is very efficient in minimizing the probability for a certain piece to become extinct and very effective in providing peers with rare pieces that can use in the Tit-for -Tat mechanism(in order to download the pieces from the peers),it miserably in case of time sensitive traffic. The reason is that with the time sensitive data each piece should be received within a certain time limit. After this deadline, piece is not useful and will be discarded. This factor is not taken into consideration in original piece selection mechanism services, since pieces are requested based on their rareness and not by their deadline. Consequently, the current piece selection mechanism needs modifications in order BT to be able to support this kind of service.

2.2 OUR APPROACH

Highest Priority Block Download First(HPBDF)- While, the rarest blocks are the most urgent ones in the traditional BitTorrent file sharing system, they are possibly not the most urgently needed one for the streaming service. For these peers enjoying view-as-download , the blocks close to the play position are the highest priority blocks, which are much urgent than the later blocks in the future. On the other hand the highest priority blocks in whole network are the one required by the majority of the peers in the network. This concept is similar to Rarest Block download first approach of BitTorrent system. The details of definitions are given below-

Highest Priority Blocks of the peers- The definition of highest priority block is different between the streaming application and the purely file downloading by the BitTorrent system. When a peer is using streaming, its highest priority blocks are those near the play position. In our work we form a set of peers to form a highest priority block set of size k, where k will have a value of 32. Mean while for a peer Downloading files by traditional BT, all of not downloaded blocks are defined as highest priority blocks and are treated equally.

Highest priority blocks of the whole system- the blocks most frequently appearing in the highest priority block set of the peers in the system are defined as the highest-priority blocks of the whole system, which means they enjoy the highest priority by majority of peers, and need to be transferred as soon as possible. In traditional BitTorrent system, *highest priority block* is the same thing as the rarest block. So we can say that concept is an extension of the original BitTorrent system. It is necessary to distinguish between the requirement of a single peer and the whole P2P system. We propose a highest priority block download first (HPBDF) to achieve trade off of benefit between the whole system and a individual peer. The detail of algorithm about how to select a block to download for a peer is given the figure 1. In our work, we have classified the blocks of a peer as follows:

Downloaded Blocks- The blocks successfully downloaded from the network.

Downloading Blocks- The blocks being downloaded.

Not requested Blocks- The blocks never requested before.

Not downloaded blocks- The blocks not available in a peer including the downloading and not requested blocks.

HPBDF-With probability p, the peer selects the highest priority of itself to downloads (line 4 of fig.1), and with (1-p) probability it downloads some high priority blocks of its neighbors which can help it to improve the ability to contribute the system to gain more reputation (line 5 of figure1). HPBDF achieves a nice balance between the streaming performance of single peer and contribution to overall network. Here p is probability having distribution function similar to zipf function [3]. Probability p is proportional to $1/ (q+1-q^*)^\alpha$, where q is the index of the block, and q* is the index first not downloaded block. Parameter p and α can be tuned so that policy is more or less aggressive with respect to their performance to earlier Pieces. HPBDF is resilient to the flash crowded scenario, where a lot of peers join at the same time to request Streaming service.

For each peer Pi, whose neighbor set is Pi.N_B, it makes a decision to select a block to download as follows:

```
1. Refresh the highest priority block set HPB(j)   of each neighbor Pj in Pi..N_B. The peer P assembles all of HPB(j)
to form highest priority block set of it neighbors,{HPB(j) | PjЄPi.N_B}.

2. Refresh the current total no of downloading blocks μ;
3. If P<h(h is constant threshold value) then
4. With the probability p, rank the not requested blocks by their played time, choose the first the first one of them to
download.
5. With the probability (1-p), download the block with highest frequency appearing in the {HPB(j) | PjЄ | Pi.N_B}. If
more than one block has highest frequency choose the block randomly from then.
6. End if
7. Sleep for an interval then GOTO 1.
```

Fig.1: Highest Priority Block Download First Scheme (HPBDF)

The highest priority blocks of most of the peers are similar. At that time there will be many blocks having the same highest frequency appearing in {HPB(j)ǀPjЄ Pi.N_B}. HPBDF chooses to randomly download the blocks with the highest frequency(line 5), which will direct the different peers to fetch different blocks most wanted by them, so that they can cooperate to deliver their highest priority blocks as soon as possible.

Modified Incentive Mechanism (MIP)- Client of traditional BitTorrent divides the neighbors into two categories chocked/un-choked [1] and only upload data to un-choked peers. At each scheduled interval (normally10 seconds), a client ranks its neighbor by their speed and puts the top peers into the un-chocked set. It also picks out the some ones randomly from the left choked peers (optimistic un-choke)[1] to change them un-chocked , so that new joined peers can gain the chance to download the data. This is called tit-for-tat incentive strategy in Bittorrent protocol to encourage peers to contribute more upload bandwidth. Although incentive mechanism is very effective in case of file downloading. We present the modified incentive mechanism (MIP) to make data flow more suitable for streaming service. The main idea of the MIP is to set all the neighbors of any peer P(i) un-choked, and define upload priority for each neighbor pj as Up. This idea is similar to [10], but they have considered only the peer in leecher state. In fact they have propose their solution on the basis of old BitTorrent protocol. In new BitTorrent peers behaves differently in both the leecher and seeder state. MIP is divided into two states.

1. Leecher State
2. Seeder State

Leecher State- Leechers are those peers which do not have complete file. In leecher state peers un-choke the Neighboring peer on the basis of their upload speed. For each neighbor Pj of the peer Pi, the initial Up is set to 1. v`(pj) is the speed of Pj in the last round. The procedure to refresh Up is as follows;

```
1. Refresh the current speed v(pj);
2. If pi is in the chocked set of pj then
3. Up= Up*(1+a);
4. else if v(pj) > v`(pj) then
5. Up = Up*(1+a);
6.else if v(pj) < v`(pj) then
7.Up = Up*(1-b);
6. end if
7. v`(Pj) = v(Pj);
8. Sleep for an interval then GOTO 1.
```

Fig.2: Modified Incentive Mechanism(MIP) in leecher state

Seeder State- In this state peer un-choke the neighboring peers not on the basis of their upload speed but the last time they were un-choked. We have considered this fact also in MIP which is totally neglected in [10]. Peers that are kept un-choked according to the time they were last un-choked the seed kept un-choked (SKU) peers, and the un-choked peer selected at random(SRU) peer. With this new algorithm, peers are no longer

un-choked according to their upload rate from the local peer, but according to the time of their last un-choke. As consequence, the peers in the active set are changed regularly each new SKU peer taking an unchoke of the oldest SKU peer. These facts are taken care in our algorithm, which were totally neglected in [10]. Our algorithm takes advantage of this new change in tit-for-tat incentive mechanism. Details of this algorithm are as follows:
For each neighbor p of peer Pj, the initial Up is set to 1.t`(pi) was the last time the peer Pj was un-chocked. The procedure to refresh pj is given below:

```
1. Refresh the current un-choke time for the peer pj;
2. If pi is in the chocked set of peer pj then
3. Up = Up*(1+a);
4. else if t(pj) > t`(pj) then
5. Up = Up *(1+a);
6. else if t(pj) < t`(pj) then
7. Up = Up* (1-b);
8. end if
9. t`(Pj) = t(pj);
10. sleep for an interval then go to step 1.
```

Fig:3 Modified Incentive Mechanism in Seeder State

Values of a, b are 0.2, 0.1 respectively, which is similar to the research work in [10]. In study [11] Piatek et al. present a new mechanism to make use of tit-for-tat to improve the download rate greatly.

3. Contributions and Future Work

We have proposed two algorithms for incorporating streaming service in BitTorrent protocol. Along with the modified piece selection policy we have proposed a modified incentive mechanism. We have covered the behavior of incentive mechanism in both leecher state and seeder state. In our future work, we will verify our proposal using simulation. Finally, we plan to incorporate our modification in BitTorrent Mainline client [12] and evaluate our work in Planet lab [13].

4. References

[1] B. Cohen. Incentives build robustness in BitTorrent. In first workshop on the economics of Peer 2 Peer systems, Berkley CA June, 5-6-2003.
[2] A nnapureddy, S., Gkantsidis, C., Rodriguez, P. R.: Providing Video-on-Demand using Peer-to-Peer Networks, Proc. Workshop on Internet Protocol TV (IPTV) '06, Edinburgh,Scotland, May 2006.
[3] Niklas Carlsson and Derek L. Eager Peer-assisted on demand streaming of stored media using Bittorrent like protocols in Proc IFIP/TC6 '07, Atlanta.
[4] K.N Parvez, C. Williamson, Anirban Mohanti and Niklas Carlsson, Analysis of BitTorrent like protocols for on deamand stored media streaming in ACM SIGMETRICS '08, June 2-6 2008, Annapolis, Maryland, USA.
[5] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In Proc. ACM SIGCOMM '04, pages 367–378, Portland, OR, August 2004.
[6] Hafeeda and B. Bhargava, "Promise:Peer-to-peer media streaming using collectcast," in ACM International Conference on Multimedia, Berkeley, California, USA, Nov. 2003.
[7] C. Dana, D. Li, D. Harrison, and C. Chuah. Bass: BitTorrent assisted streaming system for video-on-demand. In International Workshop on Multimedia Signal Processing (MMsP) IEEE Press, 2005.
[8] X. Zhang, J. Liu, B. Li, and T.P. Yum. Coolstreaming/donet: A data-drivenoverlay network for peer-to-peer live media streaming. In Proceedings of IEEE/INFOCOM, Miami, March 2005.
[9] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A.E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In Proceedings of IPTPS, Ithaca, New York, Feb 2005.
[10] Jianming LV, Xueqi Cheng, Tienying Zhang, Simming Lin, Qing Jiang, Lei Wang, Jing ye, Live BT: providing video ondemand streaming service over BitTorrent system, 2007 IEEE, Eight International Conference on Parallel and Distributing Computing Applications and Technologies.
[11] M. Piatek, T. Isdal, and et al. Do Incentives Build Robustness in BitTorrent? In *USENIX NSDI'07*.
[12]The Official BitTorrent Home Page. http://www.bittorrent.com/.
[13] The PlanetLab project. http://www.planet-lab.org/.
[14].PeerCast. http://www.peercast.org/.
[15].Streamer P2P. http://www.streamerp2p.com/
[16].P2P-Radio. http://p2pradio.sourceforge.net/.