2010 16th International Conference on Parallel and Distributed Systems

Discovering Free-riders Before Trading: A Simple Approach

Raymond Lei Xia
UFIDA (Hong Kong) Co. Ltd.
Central, Hong Kong SAR, PRC
e-mail: raymondhahk@gmail.com

Jogesh K. Muppala
Dept. of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong SAR, PRC
e-mail: muppala@cse.ust.hk

***Abstract*—Free-riding is one of the most serious problems encountered in Peer-to-peer (P2P) systems like BitTorrent. Incentive mechanisms, including those based on reputation have been proposed to deal with this problem, but are still not effective in preventing free-riders from completing a download. This is because they discover the free-riders' behavior during or after the process of trading, giving free-riders the opportunity to download from others. In this paper, we propose PreDiscover, a novel approach to prevent free-riding behavior in BitTorrent. In PreDiscover, regular peers and free-riders can be recognized before trading. So free-riders have little opportunity to download blocks from others. Our simulation results indicate that this new mechanism is very effective in discouraging free-riders and foster fairness.**

Keywords-Free-riding; P2P; Reputation System

I. INTRODUCTION

Peer-to-Peer (P2P) file sharing systems have proved to be effective for designing scalable and robust mechanisms for delivering large files to several users. Peers in the system not only download files from the server, but also share content with each other, alleviating the server's burden. BitTorrent (BT), one of the most popular P2P applications, employs incentive mechanism like tit-for-tat (TFT) to encourage contribution and discourage *free-riders*. A free-rider is a peer that downloads without a corresponding upload contribution. The presence of free-riders has a significant impact on the overall system performance. Some papers [2] [10] claimed that BitTorrent's incentive policy works well to dissuade free-riding behavior. However, others [1] reported that TFT is not as effective at discouraging free-riders. Reputation [11] based approaches to discourage free-riders get peers' reputation values after trading with them. So free-riders can still get downloading opportunities during the trading.

In this paper, we propose the PreDiscover algorithm that helps distinguish free-riders from benign peers even before trading. The only opportunity that free-riders get to download from other peers is when they newly join the system. Therefore, a free-rider is always choked by regular peers after becoming a non-newcomer. We show using simulations that this effectively discourages free-riders.

This paper is structured as follows. We first give a brief overview of the mechanisms in BitTorrent and how free-riders can exploit the mechanisms to their benefit in Section II. We then present the details of the PreDiscover algorithm in Section III. We review our simulation settings in Section IV. Thereafter we present several simulation results illustrating the benefits of using PreDiscover, in Section V. We show how an improved PreDiscover algorithm can deal with malicious peers in Section VI. We give a brief background of related work presented in the literature in Section VIII. Finally we present the conclusions of the paper.

II. FREE-RIDING IN BITTORRENT

A. Mechanisms in BitTorrent

BitTorrent mechanisms mainly consist of *peer* and *piece selection strategies*. A good peer selection strategy should maximize the service capacity of the system, and an efficient piece selection strategy should guarantee that each peer can find *interesting* pieces from its neighbors. A detailed description of these mechanisms can be found in [6][12]. Here we briefly summarize these mechanisms for completeness so that further discussions in the subsequent sections can be placed in proper context.

The peer selection strategy uses four main mechanisms: tit-for-tat (TFT), optimistic unchoking (OU), anti-snubbing, and upload only. Summed up together, they form the basis for the *choking algorithm* used by a peer. The major aim of these mechanisms is to improve the downloading experience of those peers that contribute to the file exchange, and punish free-riders. When a peer wants to download pieces from its neighbors, it adopts piece selection strategies that include the following four mechanisms: Strict Priority, Rarest First, Random First Piece, and Endgame Mode.

B. Mechanism used by Free-riders

Free-riders exploit some mechanisms of BitTorrent for their own benefit. In the original BT, free-riders can get benefits from both seeds and regular peers. On the one hand, seeds upload to requesters based on the requesters' downloading rate instead of uploading rate because seeds do not need to download any pieces. Thus free-riders can exploit this mechanism to download from seed if the free-riders have good download bandwidth. Similarly, free-riders can obtain blocks from the regular peers because of the optimistic unchoking policy [6]. Under this policy, peers would upload blocks to a node for about 30 seconds even if they receive nothing in return from that node. In the following section, we describe how we attempt to plug these loopholes by our new algorithm.

III. PreDiscover Algorithm

Free-riders always choke other peers regardless of the bandwidth. PreDiscover algorithm is based on a peer using this characteristic to discover free-riders amongst all its neighbors. A peer records the status of its neighbors, whether they are well-behaved, or are exhibiting free-riding behavior. Each neighbor then shares its collective view about its neighbors with all its neighbors. Through the exchange of this information (gossiping), peers are able to pinpoint and isolate the free-riders by denying them opportunities to download.

A. *Neighbor Status in PreDiscover*

First, every peer maintains a status vector, recording its current view of the neighbor's free-riding behavior. Each element of the status vector is a tuple (NC,FR), where NC records whether the peer is a new comer; FR represents whether the peer is a free-rider, each being either 1 or 0. The three possible values that the tuple can take are: (0,0), (1,0) and (1,1), status (0,0) means the peer is a newcomer and a potential free-rider; (1,0) means the peer is a non-new comer and a potential free-rider; (1,1) means the peer is not a newcomer and is a regular peer. PreDiscover by default assumes that every node is a free-rider (guilty until proven innocent). It is up to each node to establish to its neighbors that it is not a free-rider by exhibiting good neighborly behavior by uploading to the neighbors in return for the download that it obtains from them. Any node that does not reciprocate others' generosity will obviously continue to be treated as a free-rider. When a peer joins the swarm for the first time, it is considered to be in status (0, 0). In this state, we allow the newcomer the benefit of doubt and let it download opportunistically from others just so that it can bootstrap itself into the swarm. Once a node uploads one or more blocks to a newcomer, then the node will change the status of the newcomer from (0,0) to (1,0) in its *status vector table*. When a peer unchokes other peers, its status will be considered as (1,1) by the peers it unchoked. The three status transformations are shown in Fig. 1.

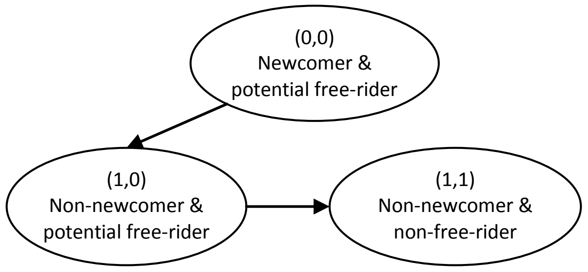

Figure 1. Status of Peer.

B. *Information Propagation*

When two peers meet, which means one is in the other's neighbor set, they exchange their status vectors, and compute the "global" status vector. Let $S_i[j] = (NC_i^j, FR_i^j)$ represent the status vector entry for node *j* in node *i's* table. For the status propagation, PreDiscover uses the following equation:

$$\forall_j \quad \forall_{k \in Neighbor(i)} \quad NC_i^j = NC_i^j \parallel NC_k^j \quad (1)$$

$$\forall_j \quad \forall_{k \in Neighbor(i)} \quad FR_i^j = FR_i^j \parallel FR_k^j \quad (2)$$

where "$\parallel$" is the OR logical operator. Peer *j* is a peer in the swarm, and peers *i* and *k* are neighbors of peer *j*. The PreDiscover choking algorithm is described in Fig.2.

```
Choking Algorithm from the perspective of peer A
for each peer i in the neighbor set of peer A
  Peer A and Peer i share the status vector
end for
for each peer B in the neighbor set of peer A
  if A is a free-rider
    do choking
  else /* A is a regular peer */
    if peer B in status (0, 0)
        use TFT or optimistic unchoking
        update the status of peer B to (1,0)
    else if peer B is in status (1, 0)
        do choking
    else if peer B is in status (1, 1)
        use TFT or optimistic unchoking
    end if
  end if
end for
```

Figure 2. PreDiscover Choking Algorithm.

C. *PreDiscover Process*

First of all, suppose peer *A* is a regular peer, then it follows the following steps:

1. Peer *A* joins a swarm; it is in status (0,0) from the perspective of all the other nodes. Peer *B* and *C* are neighbors of peer *A*, and both peer *B* and *C* use TFT and OU on peer A because $S_B[A]$ and $S_C[A]$ are (0,0),
2. Once *A* is unchoked by *B* by OU, *A* is in status (1,0) in *B*'s view, which means $S_B[A]$ is (1,0) and $S_A[B]$ is (1,1). Therefore, *B* will consider *A* as a potential free-rider and choke *A* until $S_B[A]$ is (1,1). However, *C* may still use TFT and OU on *A* since $S_C[A]$ is still (0,0).
3. $S_C[A]$ will be updated to (1,0) when *C* exchanges its status vector with *B*. Then, $S_B[A] = S_B[A] \parallel S_C[A]$ from the view of *B*, and $S_C[A] = S_B[A] \parallel S_C[A]$ from the view of *C*. According to equations (1) and (2), both $S_B[A]$ and $S_C[A]$ will become (1,0). Therefore, both *B* and *C* consider *A* as a potential free-rider and choke *A*.
4. When *A* unchokes *B*, *A's* status is updated $S_B[A]$ = (1,1) by *B*. Then, *B* will know that *A* is not a free-rider, and will use TFT and OU on *A*. However, *C* still considers *A* as a potential free-rider as $S_C[A]$ is still (1,0).
5. When *B* and *C* exchange again, $S_C[A]$ will be updated to (1,1). Therefore, both *B* and *C* consider *A* as a regular peer and use TFT and OU on *A*.

6. When A completes downloading all the blocks of the file, it may either stay on as a seed or leave the swarm.

When we consider the case that A is a free-rider, the first three steps are the same as that of the regular peer. However the remaining steps are as follows:

4. After getting some blocks from B, A never unchokes other nodes. Therefore, both $S_B[A]$ and $S_C[A]$ will always be (1,0), and both B and C consider A as a potential free-rider and choke A forever.
5. A cannot get any more blocks either from B or C, so it will not be able to complete downloading the file.

First, the algorithm addresses the problem of free-riders getting blocks from regular peers using OU, by completely removing this avenue of exploit. On one hand, a regular peer is considered as a good peer (status (1,1)) by some neighbors, so they continue to trade with it using either TFT or OU policy. On the other hand, a regular peer may still be considered as a "potential free-rider" by other neighbors, and may be temporarily choked by them. However, the peer's good status will ultimately be established among many others after several rounds propagation. Since a free-rider never unchokes other peers, so it is considered as "potential free-rider" forever, which means it will always be considered in status (1,0) by its neighbors, and will be choked forever.

Seeds are a special case in BitTorrent since they do not need to download from others. The only issue is whether the seed capacity can be exploited by free-riders. In PreDiscover we consider two possibilities: (1) Seeds in PreDiscover adopt a policy similar to regular peers in dealing with free-riders, i.e., a seed uploads to a peer only if it is status is (0,0) or (1,1) from the seed's view; or (2) Seeds adopt a free-rider agnostic view whereby they upload to any peer without discrimination. Obviously in the second case the free-riders still have a chance of completing the download, albeit at a slow pace. We demonstrate this through some experiments.

IV. Experimental Evaluation

A. *Experimental Settings*

In this part we describe the settings used to conduct our simulation experiments. We also provide details about the performance metrics that we used to evaluate the various mechanisms.

We use the BitTorrent simulator BTSim [1] to conduct the simulation experiments. We modified the simulator to implement the algorithms described earlier. We use a file size of 100 MB. We simulate a heterogeneous environment and set the download/upload bandwidth (kbps) of participate peers as the following three groups: 1500/400, 784/0 (free-riders), 784/128 and each of the groups contains one third of the total number of peers. Unless specified otherwise, the results in next section are corresponding to the setting in Table 1.

TABLE I. Experiment Settings

File Size	100MB
Block Size	256KB
Leecher Unchoking	5
Seed Unchoking	5
Percentage of free-riders	1/3

B. *Metrics*

We evaluate the system's performance using the following metrics:

Fairness: An effective P2P system should foster fairness in terms of the blocks downloaded and uploaded by peers, implying no peer in the system should download more than it uploads. We compute fairness as the maximum number of blocks served over the total number of blocks received. Obviously, when the fairness value is much larger than 1, it indicates that some peers uploaded more than they downloaded, so the system is unfair. So the closer to 1 the fairness value is, the better the system performs.

Scalability: We evaluate the system's scalability by evaluating the percentage of nodes that finish downloading as a function of time with the increase of the percentage of free-riders in the system.

Downloading time: we evaluate the time for completing the download for regular peers in both original BT and PreDiscover. The downloading time indicates the effectiveness of the system at encouraging contribution and effectively discouraging the free-riders.

V. Experimental Results

In this section, we show the experimental results for fairness, downloading time and scalability to evaluate the system's performance. In addition, we compared PreDiscover with other reputation based systems like One-hop Reputation [11].

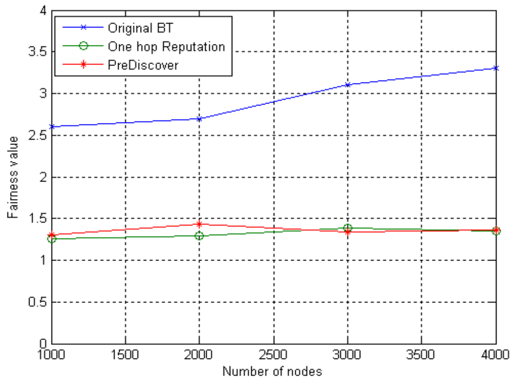

Figure 3. Fairness.

A. *Fairness*

As we can see from the Fig. 3, some peers upload more than three times the total number of blocks that they download in original BitTorrent. The unfairness is caused by the Optimistic Unchoking and the TFT protocol, which give opportunities for free-riders to exploit the regular peers.

Both PreDiscover and one-hop reputation yield a fairness value much closer to 1. However, the fairness value is a little more than 1 because free-riders still manage to download some blocks when they are newcomers. Even with One hop Reputation peers choose neighbor with high reputation value, so free-riders are discouraged, but are not totally prevented from finishing their download.

B. *Downloading Time of Regular Peers*

We evaluated the effect of the algorithm on the downloading time of regular peers. We considered a system of 8000 peers with one third peers as free-riders. Similar trends were observed for different number of peers.

As we can see from Fig. 4, regular peers' downloading time in One hop Reputation has only improved a little compared with those in Original BT, but regular peers in PreDiscover perform much better than those in both Original BT and One hop Reputation. In addition, the downloading time is much shorter in PreDiscover. The reason is that free-riders are better identified and shunned in PreDiscover.

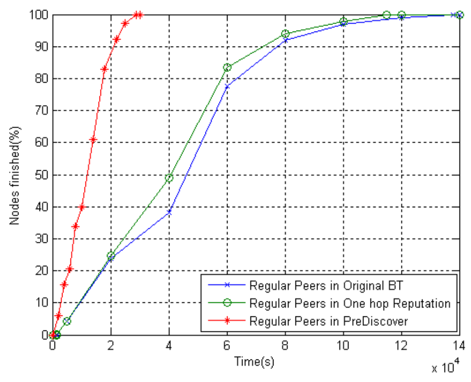

Figure 4. Behavior of Regular Peers.

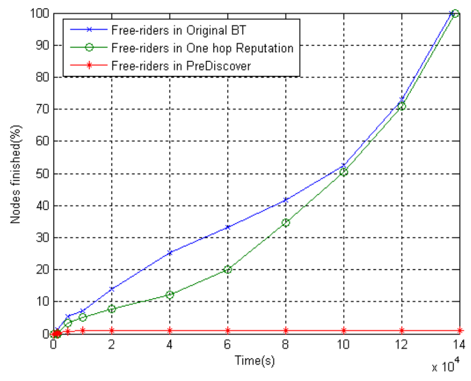

Figure 5. Behavior of Free-riders.

C. *Downloading Time of Free-riders*

Next, we compare the free-riders' behavior in PreDiscover, One hop Reputation and Original BT. Since free-riders in PreDiscover cannot finish their downloading, so we use the percentage of blocks finished instead of nodes finished vary with the time. As we can see from Fig. 5, free-riders in One-hop Reputation perform similar to those in Original BT, and finish their downloading finally, which means One hop Reputation cannot prevent free-riders completing the download. However, in PreDisocver, free-riders download less than 1% of total blocks and never finish downloading. This is because free-riders are choked after they download a few blocks, and thus effectively prevented from downloading.

D. *Robustness and Scalability*

Next, in order to see the robustness and the scalability of PreDiscover, we set the percentage of free-riders to 5%, 10% and 33% for both PreDiscover and One hop Reputation, and get the following results. As we can see from Fig. 6, with the increase of the percentage of free-riders, One-hop Reputation performs worse on node downloading times. However, in PreDiscover, the results of the three settings are similar. This is because the free-riders are discovered and effectively shunned. Thus, PreDiscover is robust and scalable even with increasing number of free-riders.

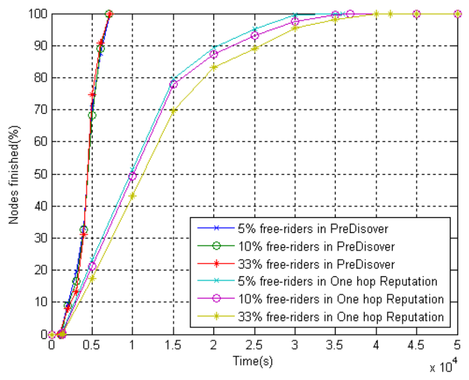

Figure 6. Scalability and Robustness of PreDiscover.

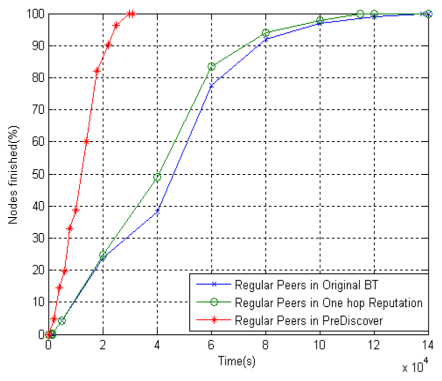

Figure 7. Regular Peers' Behavior for Seeds Not Using PreDiscover for 8000 nodes

E. *Seeds Not Using PreDiscover*

In this section, we consider the case where seeds do not use PreDiscover, and a seed uploads to peers based on the peer's download rate. We use the same setting as the previous experiments, and we set the total number of nodes to 8,000. Fig. 7 shows that regular peers' behavior is very similar to the results earlier. The reason is that regular peers' download time mainly depends on the collaboration between regular peers rather than the seed, so a seed has a little influence on the regular peers' download time. However, this has a big influence on free-riders. As we see from Fig. 8, free-riders can complete the download, but the download time has been delayed significantly compared with the case in Original BT and One-hop reputation algorithm.

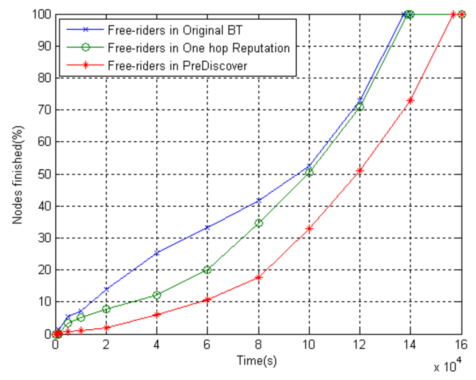

Figure 8. Free-Riders' Behavior for Seeds Not Using PreDiscover for 8000 nodes

VI. Improved PreDiscover for Malicious Peers

A. *Malicious Peers*

So far we assumed that peers are either regular peers or free-riders. We notice that the earlier algorithm classifies peer as belonging to one of the two classes. Once a peer is classified as a regular peer, it enjoys all the benefits of receiving upload from others. In this section we deal with the case of a Malicious Peer. Specifically we consider the case where the peer could fool others to consider it to be a regular peer by just uploading a few blocks in the beginning until it gets classified as a regular peer at least by a few other peers, and then stops uploading. In PreDiscover, status (1,1) is the final status. Once a malicious peer uploads just a few blocks to others, it is considered to be in status (1,1) and remains there forever. Therefore, the malicious peer can download freely from others after a small contribution. The presence of the malicious peers has a bad influence of the system's performance, as expected. We extend our PreDiscover algorithm to deal with this problem. We do not explicitly deal with colluding attacks, as it requires significantly more effort to identify and defeat.

B. *Improved PreDiscover*

In our Improved PreDiscover algorithm, peers are still assigned one of three statuses: (0,0), (1,0) and (1,1). The difference is that state (1,1) is no longer the final state. When a peer *A* is considered to be in state (1,1) by peer *B*, then *B* uses the TFT and OU on *A* for 10 seconds (this is configurable in the system settings), then change *A's* status back to (1,0) in *B's* state vector table. If *A* just upload to *B* a few blocks and does not upload any more, *A* can only get served for 10 seconds and will be considered to be in status (1,0) thereafter and will be choked forever by *B*. Thus a peer needs to periodically re-establish its *good-neibhorliness* by uploading to its neighbors. Fig. 9 shows the state transition diagram. The algorithm is shown in Fig. 10.

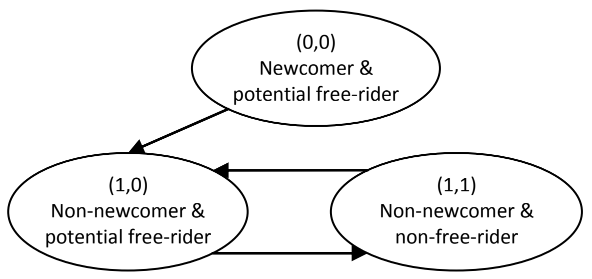

Figure 9. Status of Peers in Improved PreDiscover

```
Choking Algorithm from the perspective of peer A
for each peer i in the neighbor set of peer A
  Peer A and Peer i share the status vector
end for
for each peer B in the neighbor set of peer A
  if A is a malicious peer
    upload a few blocks and do choking
  else /* A is a regular peer */
    if peer B in status (0, 0)
        use TFT or optimistic unchoking
        update the status of peer B to (1,0)
    else if peer B is in status (1, 0)
        do choking
    else if peer B is in status (1, 1)
        use TFT or optimistic unchoking for a few seconds
        change peer B bacl to status (1,0) in state vector
    end if
  end if
end for
```

Figure 10. Improved PreDiscover Algorithm

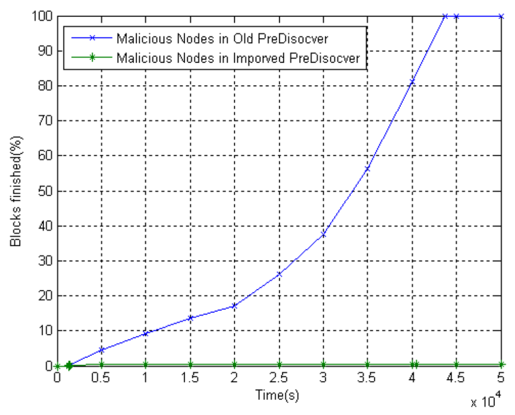

Figure 11. Malicious Node performance in PreDiscover and Improved Prediscover.

C. *Experimental Results*

We use similar settings as the previous experiments, and we set the number of nodes to be 1,000, with one third of them being malicious nodes. We trace the malicious nodes' behavior for both PreDiscover and the Improved PreDiscover and obtained the results shown in Fig. 11.

The blue line in Fig. 11 shows that the malicious nodes can finish their download in PreDiscover based system. However, as the red line shows, the malicious peers can only get a few blocks and cannot complete their download in the Improved PreDiscover based system. This is because the malicious peers are identified in Improved Discover and are choked by their neighbors. Thus they cannot get any benefits from their neighbors except when they are newcomers and when they upload just a few blocks.

VII. Related Work

Several studies about the incentive mechanisms of P2P systems [12] have been published in the literature. Jun et al. [8] analyzed the original incentive mechanism of BitTorrent using PlanetLab [9]. They found that the free-riders are not punished properly, and the peers who contribute to others are not rewarded appropriately. To address this problem, they proposed a new mechanism with a deficit factor that accounts for how much a peer uploads without expecting a reciprocal download. The factor determines the amount that a peer is willing to risk for a chance to establish cooperation. Peers upload evenly to all links as much as they can under this condition. Garbacki et al. [7] proposed a novel mechanism, in which the resource, upon which incentives are built, is bandwidth rather than content. Bandwidth is unrelated to the interests of peers, so it is more suitable to be a trading unit. In the bandwidth-exchange incentive mechanism, any peers can help others to download with their idle bandwidth, regardless of their content. Chow et al. [4] presented a novel approach from the perspective of use the seed capacity appropriately with the goal of reducing the free-riders. As illustrated in some measurements and simulation results, the leechers' downloading rate is slower at the beginning when they have few chunks to exchange with others; also, it can be slower at the end due to it is hard to find peers with the few missing chunks. So the authors proposed a simple method to prioritize the use of seeding capacity to certain portions of a file downloading process. They used two ways to choose neighbors to unchoke: (1) Sort-based: a seed sorts its neighbors according to the number of chunks each has, then it unchokes N of them based on the sorting order; (2) Threshold-based: a seed unchokes N neighbors with [0..K/2]% or [(100-K/2)..100%] of the chunks. Their experimental results show that the approach of better utilization the seed capacity not only discourages free-riders' behavior, but also improves the performance of contributing leechers.

Some researchers proposed reputation-based systems in building fairness into P2P downloading. Piatek et al. [11] present a method called One-hop Reputation, in which peers maintain a persistent history of interactions to foster persistent contribution incentives. The main idea is that it restricts the number of amount of indirections between sending and receiving peer to at most one level of intermediaries. The One hop Reputation limits the propagation of information and allows for local reasoning about the trustworthiness of intermediaries, therefore it fosters scalability and robustness.

VIII. Conclusions

In this paper, we proposed a novel incentive method named PreDiscover. The main difference between PreDiscover and the other reputation methods is that peers in PreDiscover System know that whether their neighbors are free-riders or regular peers before trading with them, and it is helpful to make correct decision on selecting neighbors to trade with. Therefore, this new approach gives few opportunities to violate peers to download from others without any contribution. We implemented PreDiscover in BitTorrent simulator and the results shows that PreDiscover can prevent free-riders effectively.

References

[1] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, "Influences on cooperation in BitTorrent communities," Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, 2005, pp. 111-115

[2] A. Bellissimo, P. Shenoy, and B. N. Levine, "Exploring the use of BitTorrent as the basis for a large trace repository," Technical report, University of Massachusetts Amherst, Dept. of Computer Science, June 2004

[3] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Network's Performance Mechanisms," IEEE INFOCOM, April 23-29, 2006, Barcelona, Catalunya, Spain, pp. 1-12.

[4] A. L. H. Chow, L. Golubchik, and V. Misra, "Improving BitTorrent: A Simple Approach," The 7th International Workshop on Peer-to-peer Systems (IPTPS'08).

[5] B. Cohen, http://www.bittorrent.com/.

[6] B. Cohen, "Incentives Build Robustness in BitTorrent," In Proc. First Workshop on Economics of Peer-to-peer Systems, Berkeley, USA, June 2003

[7] P. Garbacki, D. H. J. Epema, and M. van Steen, "An amortized tit-for-tat protocol for exchanging bandwidth instead of content in P2P networks," First International Conference on Self-Adaptive and Self-Organizing Systems (SASO), July. 2007, pp. 119-128

[8] S. Jun and M. Ahamad, "Incentives in BitTorrent Induce Free Riding," Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, Philadelphia, Pennsylvania, USA, August. 2005, pp. 116-121.

[9] http://www.planet-lab.org/

[10] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. Al Hamra, and L. Garc´es-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," Passive and Active Measurements (PAM), vol. 3014, Apr. 2004, pp. 1-11

[11] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and T. Anderson, "One hop reputations for peer-to-peer file sharing workloads" In NSDI, 2008.

[12] L. Xia and J. Muppala, "A Survey of BitTorrent performance", *IEEE Communications Surveys and Tutorials*, Vol. 12, No. 2, Second Quarter 2010, pp. 140 - 158.