Peer-to-Peer Netw. Appl. (2013) 6:86–100
DOI 10.1007/s12083-012-0141-y

Securing BitTorrent using a new reputation-based trust management system

Behrooz Shafiee Sarjaz · Maghsoud Abbaspour

Received: 29 September 2011 / Accepted: 2 May 2012 / Published online: 17 May 2012

Abstract Nowadays, BitTorrent as a means of sharing files has become highly popular among internet users. However, due to the open nature of BitTorrent protocol and lack of any security mechanism, number of attacks against BitTorrent has significantly increased. Sybil, Collusion, Lying-Piece, Fake-Block, and Chatty-Peer are attack types which have been considered in this paper to secure BitTorrent against them. These attacks can decrease the download performance of BitTorrent clients considerably. In this paper a new reputation based trust management system to cover aforementioned attack types is presented. The proposed approach calculates a local score at peers and a global score at the tracker for each peer. First, peers are sorted according to their cumulative score at the tracker and then top 10 % of these peers are used to determine other peers global score. These local and global scores are used to find attackers. In addition, a novel formula has been utilized to calculate peers local score. Using the global score concept makes our mechanism robust and swift to detect collusion attack which has not been considered in most of similar previous works. In order to evaluate the effectiveness of the proposed system, several simulation and real experiments in the Emulab testbed were performed. The outcomes indicate that our method is highly effective in detection of rogue peers and Free-Riders; moreover, performance of honest peers has significantly improved.

B. Shafiee Sarjaz · M. Abbaspour (✉)
Department of Computer Engineering, Faculty of Electrical and Computer Engineering,
Shahid Beheshti University, G. C., Evin.,
Tehran, Iran
e-mail: maghsoud@sbu.ac.ir

B. Shafiee Sarjaz
e-mail: behrooz.shafiee@ieee.org

Keywords Peer to peer security · BitTorrent security · Reputation based Trust Management

1 Introduction

BitTorrent [1] has become one of the most popular peer-peer (P2P) protocols. According to a recent study [2] 66 % of P2P traffic which is approximately 35-55 % of the whole internet traffic belongs to BitTorrent. BitTorrent popularity is due to its fundamental advantages over traditional client-server architecture such as: self-scalability, reliability, fairness and cost- efficiency.

Using BitTorrent as a means of sharing copyrighted contents has provoked multimedia producing companies to impede distribution of their products over BitTorrent [3]. Expectedly, multimedia industries have begun Torrent-Poisoning carried out by Anti-Piracy organizations as an attempt to prevent the peer-to-peer (P2P) sharing of copyrighted content [4, 5]. Torrent-Poisoning is the act of intentionally sharing corrupt data or using different kinds of attack to slow down BitTorrent downloads. Clients such as ZipTorrent [5] are an indication to this attempt; the goal of ZipTorrent is to slow down popular downloads as much as possible. Anti-Piracy organizations utilize hundreds of these clients at the same time and this can potentially bring the average download speed down to zero [5]. The evaluation of these kinds of attack shows that they can seriously threaten BitTorrent protocol. In [6] a mixed attack strategy is proposed which its results indicate that the robustness of the BitTorrent swarm can be damaged and most peers cannot complete the downloading even though the attack is not distributed to every downloading peer evenly. Moreover, the evaluation results of Lying-Piece attack against BitTorrent by Konrath et al. [7] indicate that this attack type can significantly damage the performance of

swarm and when number of attackers is high enough, the attackers can even cause general swarm failure.

In addition to different kinds of attacks against BitTorrent, Free-Riding is another serious threat to it. Free-Riding is profiting BitTorrent file-sharing but not contributing resources to the overall system capacity [8]. As the fraction of free-riders increases, the overall system capacity per peer decreases. Although BitTorrent incentives long have believed to be robust to strategic manipulation, recently come under inspection. Clients like BitThief [9] and BitTyrant [10] empirically show that users are not any more encouraged to follow the BitTorrent protocol. For instance, the BitTyrant client can reach relatively high download rates, surprisingly in some conditions, even a 70 % speed improvement in comparison with regular clients [10].

Based on the stated reasons, the conclusion can be drawn that existence of a defense mechanism for the BitTorrent protocol is absolutely essential. Due to open nature of BitTorrent and existence of many challenges in cooperation with unknown partners, providing a comprehensive defense mechanism has many difficulties. Sharing knowledge between peers is one of the ways to make at least some trust among peers. This system is called reputation-based trust management [11].

In this paper a new reputation-based Trust Management System (TMS) has been proposed to defeat different kinds of attacks against BitTorrent as well as improving performance of BitTorrent protocol by limiting Free-Riding. The proposed TMS profits the central point in the BitTorrent, the Tracker. Tracker is used to collect and propagate peers' experiences about each other in addition to judge about peers and recognition of misbehaving peers. The main advantages of the proposed TMS are: 1) Resistance in presence of high amount of malicious peers with collusion among them 2) Covering different attack types. 3) Easy integration with BitTorrent present protocol with a negligible overhead.

The proceeding parts are organized as follows: in section 2 related works are discussed. Section 3 presents preliminaries, then section 4 describes the proposed method, section 5 presents simulations and Emulab implementation results; finally, section 6 concludes the paper.

2 Related works

Due to the excessive amount of related works only two main groups of works are discussed; in the first group, works on BitTorrent incentive are discussed; then, Trust management works are reviewed.

2.1 BitTorrent incentive

The study of BitTorrent protocol's incentives began by its inventor, Cohen. He demonstrated that tit-for-tat-based incentives make BitTorrent robust to strategic gaming [12]. In contrast, several studies have shown that the current BitTorrent mechanism is not adequate to prevent free-riding and cannot encourage peers to cooperate enough [9, 10, 13].

Piatek et al. [10] showed that incentive mechanism of BitTorrent is not robust to strategic clients. In order to prove their claim, they developed BiTyrant which tries to determine the exact amount of contribution necessary to maximize its download rate by dynamically adapting the upload rate allocated to neighbors. The authors claim that BitTyrant provides a median 70 % performance gain for a 1 Mbps client on live Internet swarms.

Locher et al. [9] study BitTorrent's vulnerability to selfish-behavior. They showed that even entire files can be downloaded without reciprocating at all. To this end, they presented BitThief, a free riding client that never contributes any real data. BitThief connects to all the peers that it can find in order to get the maximum amount of optimistic unchokes. They also, demonstrated that a simple trick suffice in order to achieve high download rates, even in the absence of seeds.

Levin et al. [13] arrive at a conclusion that BitTorrent's incentives are not fair, even under a simple definition of fairness which tells "the more a peer gives, the more it gets". They proposed a game theoretic model for BitTorrent and within that model, they have shown that BitTorrent does not use tit-for-tat; moreover, they proposed an auction-based model which they found to be more accurate for BitTorrent. Based on their auction-based model they expressed that BitTorrent does not use tit-for-tat as widely believed but an auction to decide which peers to serve.

2.2 Trust mangement

Trust management in P2P networks can be classified into 3 main categories: credential and policy-based trust management [14, 15], social network-based trust management [16, 17] and reputation-based trust management. In this paper only reputation-based TMSs has been considered.

Peers in reputation-based systems interact with one another and assign trust values to one another. Assigned trust values are a function of local trust scores, global trust scores or a combination of the peers' global and local trust scores.

Eigen Trust Algorithm [18] stores a global trust score for each peer which is computed from scores given by all other peers and it uses pre-trusted peers concepts to avoid collusion attack in which malicious peers conspire together in order to defame honest peers. However, no mechanism for choosing these peers is presented and there is no guarantee that these peers are really trusted. Moreover, this work does not consider lying piece attack and peers who tell lye about the scores of others. Eigen trust algorithm uses matrix

calculation to compute global score which is slow especially in BitTorrent swarms which may consist of thousands of peers. In addition, it does not make any difference between fake peers who change their identity and newcomers and impose an overhead such as CAPTCHA on newcomers to stop Sybil attack. Finally, keeping a lot of old data to compute global scores introduces a significant storage overhead in populated swarms. These issues make this mechanism impractical especially in the BitTorrent network which usually has high number of peers. The study of Hu et al. [19] is another example of algorithms which stores a global trust score for each peer but unlike Eigen Algorithm, they provide a mechanism against Sybil attack; however, it still suffers from a significant overhead.

Stakhanova et al. [20] and Singh et al. [21] proposed a TMS in which each peer computes local trust score rather than a global trust score for other peers. As it is stated by Shah et al. [22], TMSs that don't utilize global trust scores have a slower convergence rate and cannot react as rapidly as they should to the changes in peers' behaviors. Moreover, the presented TMS in [21], TrustMe, uses broadcasting the query messages to query or report the reputation scores. This method will impose a lot of message in the network. In addition, when network size is very large especially in the BitTorrent swarms, it will take a long time to disseminate scores and this is why its convergence rate is slow. Consequently, this mechanism is not suitable for implementation in highly populated networks such as BitTorrent.

From another perspective, TMSs can be classified into fully decentralized and partially decentralized as well. In fully decentralized TMSs [18, 20, 21] different algorithms have been used such as pagerank [18], fuzzy logic [23] or ecological network [24] but the large amount of computation and reputation in these mechanisms bring a significant overhead and latency to the network.

In [25] an example of partially decentralized approaches which uses a credit-debit mechanism for Gnutella network has been presented. This mechanism credits the peer for serving content and debits the peer for downloading resources. In this approach a Reputation Computation Agent (RCA) at regular intervals collects reputation information from peers. The RCA maintains the transaction state of the system keeping track the full list of transactions and points to be granted for those transactions for a period of time. After each transaction, each peer reports the transaction to RCA. This large amount of transactions imposes a communication overhead on peers. Moreover, in [25], no direct mechanism for decreasing the score for malicious behavior is presented.

To the best of our knowledge, [22] is the only work which proposes a reputation based trust management scheme for the BitTorrent network. In this work, a partially decentralized reputation-based TMS for BitTorrent is presented which uses global trust scores to evaluate peers as well as their local trust scores. The tracker functionality is extended to act as a Reputation Computation Agent but the algorithm which is used to compute global scores of peers is vulnerable to collusion attack as our simulation results show. In addition, the proposed method doesn't address Chatty-Peer attack, in which malicious peers advertise ownership of many pieces but do not upload any block neither authentic nor fake one.

In this paper, like [22] we use a partially decentralized reputation-based TMS for BitTorrent with both local and global trust score concepts. However, based on the evaluation results, our mechanism is significantly more resilient to different kinds of attack especially collusion attack in comparison with [22]. In addition, the majority of previous works in P2P networks introduce a large amount of transaction among peers and message overhead to the network which can decrease the performance of peers even in small swarms. This issue makes most of them impractical in BitTorrent network; however, our mechanism has a negligible overhead which makes it practical to integrate with BitTorrent present protocol. Finally, unlike most of previous studies, in this paper free-riding and a variety of attack types are considered and evaluated in detail.

3 Preliminaries

3.1 BitTorrent overview

In this section, sufficient relevant details about BitTorrent are presented to allow us expressing our hypothesis. BitTorrent works by a group of users (called swarms) with an interest in downloading a specific file and cooperating to accelerate the process.

In BitTorrent jargon, each file is divided into some pieces (usually of the length 256 KB) and each piece is divided to some blocks (typically 16 blocks). To release a file on BitTorrent network, a specific description file (called "torrent file") is necessary which holds the general information of the file such as name, piece length, and Pieces (a string consisting of the concatenation of all SHA-1 hash values of the pieces) as well as the trackers addresses. Each swarm is managed by the trackers (the only central points in the swarm) which keep track of all peers and their activities. There are two kinds of interactions between peers and the tracker: 1) a peer asks the tracker for a list of available peers (a maximum of 50 peers is typical) and 2) peers announce periodically the tracker of their status regarding download of the file.

Using the reply from the tracker, the client connects to the peers and downloads pieces of the file from them, as well as uploading completed pieces to other peers. When a file is being downloaded, this peer is considered as a leecher.

Once the file is completely downloaded and shared, the peer is converted to a seed.

The maximum number of peers served concurrently is configurable by the user. All other peers connected to a peer (whether they are interested or not) which are not being served are said to be choked. In consequence, each client implements an algorithm to choose which peers to choke and un-choke among those connected to him over the time. The strategy proposed by BitTorrent is named "tit-for-tat", meaning that a client will preferably cooperate with the peers cooperating with him. This strategy however, if implemented strictly, would considerably slow down the insertion of newcomers into a running swarm because they do not have anything to share at the beginning. Thus, clients which have nothing to share are given three times more chances to be selected by the optimistic unchoke algorithm. Another important algorithm in BitTorrent is the piece selection algorithm which follows the Rarest First (RF) policy. This algorithm is based on the concept of choosing the pieces that are less replicated among peers at first. When there is more than one piece with the smallest degree of replication, then the choice is random.

3.2 BitTorrent attacks

There are many kinds of attacks against BitTorrent, which are frequently deployed today, including: DoS [26], Sybil [27], Collusion [26], Lying-Piece [7], Fake-Block [28] and Chatty-Peer [28] attack. In the following, six mentioned attack types are described briefly.

3.2.1 DoS attack

Denial of Service attack happens when malicious peer sends a large amount of requests or useless messages to the victim peer. As a result, bandwidth and other resources of the victim peer are used without any gain. Covering this attack type is outside the scope of a TMS and there are several approaches for encountering with this type of attack [29, 30].

3.2.2 Sybil attack

The Sybil attacker creates a large number of identities in order to become more powerful. Depending on how identities are generated, system may be vulnerable to this attack. Usually this attack type can be prevented by refusing multiple connections from a specific IP address. In addition, there are some more effective proposed defense mechanisms against this attack [31, 32].

3.2.3 Collusion attack

In this attack type, a set of malicious peers under guidance of a central peer (leader), perform a correlated attack against the system. This attack type is the most dangerous attack type because if invader peers function correctly as planned at each step, it would be extremely difficult to locate and annihilate the attackers [26]. Usually in Reputation-Based TMSs, collusion attack's goal is to decrease the reputation and score of honest peer as well as increasing score of malicious peers. This behavior finally results in: destroying fairness among peers and a false global belief about peers. Attackers achieve to this goal by spreading false information about honest peers and themselves.

3.2.4 Lying-piece attack

The goal of attacker in Lying-Piece attack is to destroy balance of Rarest First Policy by announcing ownership of a piece it does not have. This behavior will artificially increase the replication level a piece, and will reduce the actual availability level of the piece in the swarm. This attack type is necessary and also a kind of tool for other attack types because it tempts other peers into responding the attacker false advertisement. In fact, this is a prologue for other attacks such as Fake-Block or Chatty-Peer attack. Attacker can tell lie not only about a limited number of pieces but also about a great number of or all pieces [7].

3.2.5 Fake-block attack

In this attack type, attacker advertises ownership of many pieces of the file (Lying-Piece Attack) right after joining the swarm. Upon receiving this information, victim peer sends the request to the attacker and the attacker instead of sending the authentic block sends fake block. When download of all blocks of a piece completed the victim peer performs a hash check for that piece. The check will fail and all blocks of the piece should be downloaded again. Therefore, the attacker could force the victim to redownload a 256 KB piece, only by sending a 16 KB fake block. By repeating this process, attacker can make a big delay in download process of the victim peer.

3.2.6 Chatty-peer attack

In this type of attack similar to Fake-Block attack, attacker advertises ownership of many pieces but the different point is that when victim requests one or more pieces the attacker doesn't upload any block neither authentic block nor fake one. As a result, through repetition of this scenario over and over, the victim spends a considerable amount of time dealing with attackers with no beneficial consequence.

4 Proposed trust management scheme

In this section, the proposed TMS is introduced. It uses the BitTorrent peers' transactions for calculating local scores and the BitTorrent tracker to compute global trust scores. There are five main steps in the proposed TMS, which are illustrated in Fig. 1.

At the first step, peers calculate and assign local scores to one another. These local score are calculated according to the transaction history which each peer keeps for every other peer it has had dealt with. In the second step, each peer sends calculated local scores to the tracker. As it was stated in section 3.1 peers interact with tracker at fixed intervals which are called *tracker_update_intervals*. At these intervals, each leecher sends to the tracker, a list of scores for the peers with whom it had interaction. In this way the tracker gathers scores from different leechers including honest and fake ones. When a leecher becomes a seed, because it doesn't download from other leechers, it can't distinguish malicious peers from honest peers; therefore, it won't participate in this scenario anymore. In the third step, first, the tracker calculates global cumulative score of each peer, and then finds the top 10 % of peers that have the highest global scores. In the fourth step, the top 10 % of peers determine global score of other peers, and finally in the last step, these global scores are sent back to the peers along with the tracker response. Peers after receiving global scores, use them to distinguish rouge peers from honest peers in their Choking and Optimistic Unchoking algorithms.

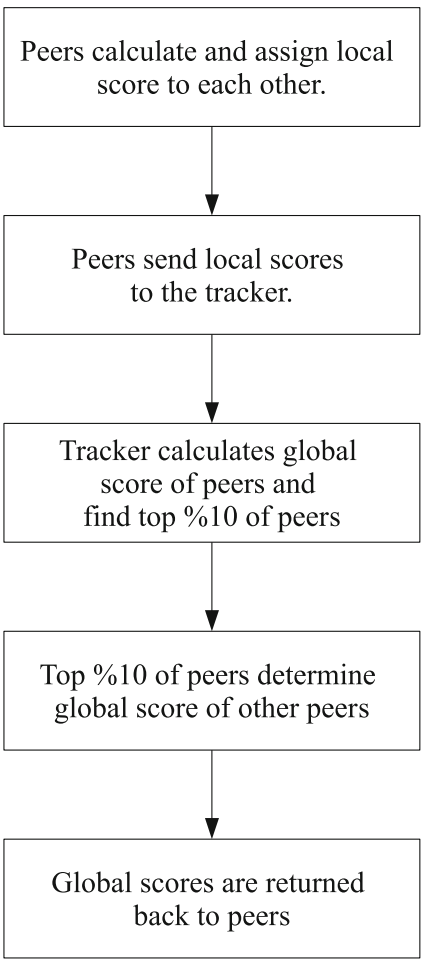

Fig. 1 The Proposed TMS main steps

As a result, during five steps, peers simple interactions are converted to valuable information for identifying malicious behaviors in a swarm. In the following sections, first trust scores calculation algorithms are presented in detail; second, robustness of the proposed TMS is discussed, and finally the modified choking algorithm is presented.

4.1 Local and global trust scores calculation

The proposed TMS utilizes both local and global information for calculating trust score for each peer. Local score calculation is done inside the peers using their transaction logs with other peers, and global scores are calculated at the tracker using peers' local scores. In the following, calculation of local and global trust score are discussed.

4.1.1 Local trust score calculation

As indicated in preceding sections, each peer keeps record of any interaction with other peers, including upload, download, unanswered request, and time of interaction. This information is kept in a local Hash Table data structure with Peer-ID as the key, to be accessed quickly. Peers can utilize this information to calculate a *fairness score* for other peers. Equation 1 defines *fairness score* of peer *j* given by peer *i*. Peer *i* calculate this score according to peer *j* recorded behaviors during recent *t* seconds as follows:

$$F_{ij}(t) = d_{ij}(t) - u_{ij}(t) - ur_{ij}(t) \quad (1)$$

Where $d_{ij}(t)$ is number of chunks that peer *i* have been downloaded from peer *j* since time *t* till now; $u_{ij}(t)$ is number of chunks that peer *i* have been uploaded to peer *j* during recent *t* seconds, and $ur_{ij}(t)$ is number of unanswered requests sent by peer *i* to peer *j* during recent t seconds. If the time passed from sending piece-request moment exceeds a threshold called *request_time_out*, then an unanswered request will be accounted.

Equation 1 considers peers' cooperation with download (d_{ij}) and upload amount (u_{ij}) parameters because for peer *i* downloading more than uploaded amount to the peer *j* shows more cooperation and more fairness score. In addition, number of unanswered requests (ur_{ij}) is considered to combat with chatty peers because more unanswered request means less *fairness score* (F_{ij}). At this point, peer *i* uses

fairness score to assign a *local trust score*, to peer *j*, with use of Eq. 2:

$$LocalScore_{ij}(t) = \begin{cases} -1 & if\ there\ is\ at\ least\ one\ bogus\ chunk \\ & in\ j's\ uploaded\ chunks \\ \left(\frac{2}{\pi}\right) * atanatan(F_{ij}(t)) & else \end{cases} \quad (2)$$

During devising Eq. 2, three major factors were considered: the first factor was fast reaction to suspicious behaviors like receiving bogus chunks. When a peer sends a bogus chunk, that peer is highly probable of being fake; therefore, upon detection of this fact, a fast reaction is required which is done with assigning the lowest possible score (−1); the second factor was necessity of a non-linear function to determine peers scores. At low fairness scores a high sensitive function is required but when fairness score is high, sensitivity should be avoided because high scores represent a steady state in peer behavior. To accomplish this concern, atan() function is used because the rate of changes in atan() function at low values is high and vice versa. As an example, by using atan() function, between local trust scores of two peers with fairness scores of −10 and −1 there is 0.43 difference but for two peers with fairness scores of −1000 and −1500 the difference is only around 0.0003. Another advantage of using atan() function is normalizing scores in the range of [−1, 1]; the third and last factor is time. Time was used in order to give a rehabilitation chance to the peers who were defamed. Sometimes due to unintentional sending of fake blocks or other uncontrollable factors such as delays, honest peers' fairness scores gets low but this should be forgotten after a period which is called *rehabilitation_interval*.

4.1.2 Global trust score calculation

At the tracker, during each update interval, global scores of peers are calculated and announced to the connected peers. One crucial point in the calculation of global score is valid both peers score each other. This means that a fake peer can't assign a negative score to an honest peer unless it communicates with that honest peer. Therefore, assigning a negative score by the fake peers to the honest peers is at the expense of being recognized by the honest peers.

In the calculation of global scores, all peers are not considered equal. First, peers are sorted descending according to their cumulative scores. Then the top 10 % of peers are selected as Super-Peer and local scores of these Super-Peers are used to calculate global score of other peers. The global score of each peer is the average score given by all Super-Peers or some of them; however, there is the possibility that all Super-Peers have no score for all peers. In addition, if the score for a specific peer is not available in none of Super-Peers, it will be the average score given by other peers. By using this mechanism collusion attacks can be controlled because this method doesn't let fake peers to affect other peers global score by trying to prevent them from being Super-Peer. The effectiveness and robustness of this method is certified in the next sections.

4.2 Robustness analysis

In this section, robustness of the proposed TMS against two different attack scenarios is discussed. The assumptions are existence of collusion attack between fake peers which means fake peers know one another, and they deliberately organize to attack the network. In addition as it was stated in section 4, in our system only bidirectional scoring is considered and unilateral scores are ignored; thus, if fake peers intend to assign negative score to honest peers they should necessarily communicate with them.

In the following, first, two different scenarios for behavior of fake peers are introduced, then essential parameters and calculation of global score of each type of peer are discussed and finally analysis of robustness in each scenario is explained.

In the first scenario which is called Active-Scenario, fake peers joins the BitTorrent swarm and communicate with other peers, assign maximum score to one another, and they also do malicious behaviors in the swarm.

The second scenario is called Passive-Scenario. In this scenario, fake peers join the BitTorrent swarm but don't communicate with other peers and they just give maximum score to one another in order to get Super-Peer positions at the tracker.

In order to analyze each scenario, four constant parameters are required. These constant parameters are:

- K_g shows average percentage which honest peers communicate with other peers.
- K_f indicates average percentage which fake peers communicate with honest peers.
- S_p demonstrates average score given to honest peers by honest peers.
- S_n shows average score given to fake peers by honest peers.

In addition to these constant parameters the only variables are number of fake peers in the swarm indicated by *F*, and the total number of peers including fake and honest peers together indicated by *N*. Calculation of constant parameters is discussed thoroughly in appendix A, also fake and honest peers global score is calculated as follows. In a swarm with total number of *N* peers including *F* fake peers and *N-F* honest peers, the global score of a fake peer is:

$$fake\ peer\ global\ score = (F-1) + K_f * (N-F) * S_n \quad (3)$$

The first part of Eq. 3 indicates the score given by other fake peers to this fake peer due to collusion assumption, and the second part of this equation is the average given score by

honest peers to this peer. The global score of an honest peer can be calculated as in Eq. 4:

$$honest\ peer\ global\ score = K_g{*}(F){*}(-1) + K_g{*}(N-F-1){*}S_p \quad (4)$$

In Eq. 4, first part is the assigned score by the fake peers to an honest peer, and the second part indicates average score given to this honest peer by other honest peers.

At this point all the prerequisites for analyzing both mentioned scenarios are presented and it's time to discuss the scenarios. In the Active-Scenario fake-peers communicate with other peers and behave according to their types. As an example, Fake-Block peers upload fake blocks and Chatty peers don't respond to honest peers requests. As a result, fake peers try to decrease performance of swarm through malicious behaviors. Fake peers follow another aim in addition to decreasing performance of swarm, they want to occupy as much as possible Super-Peer positions; thus, they give maximum scores to each other (collusion attack) and minimum score to honest peers but receive a negative score(S_n) as well.

If there are r Super-Peer positions available, then the following equation shows how many fake peers in the Active-Scenario are necessary to occupy all the Super-Peer positions.

$$\begin{array}{c} r^{th} fake\ peer\ global\ score > Max\,(honest\ peer\ global\ score) \\ r^{th}\big((F-1) + K_f{*}(N-F){*}S_n\big) > Max(K_g{*}(F){*}(-1) + K_g{*}(N-F-1){*}S_p) \end{array} \quad (5)$$

In order to calculate the score of r^{th} peer of fake peers, in case of our TMS, 10 % of total swarm peers which is $N{*}0.1$, a large amount of fake peers global score samples were collected. By plotting these scores, we observed that scores follow an arc tangent function, and score of r^{th} fake peer= $N{*}0.1$ is on average around 0.85 of the maximum score of fake peers. Therefore Eq. 5 is simplified as follows:

$$\begin{array}{c} 0.85{*}Max\big((F-1) + K_f{*}(N-F){*}S_n\big) > \\ Max\big(K_g{*}(F){*}(-1) + K_g{*}(N-F-1){*}S_p\big) \end{array} \quad (6)$$

Replacing constant parameters from Appendix A gives that, in order to satisfy Eq. 4, fake peers must consist at least %36 to %40 of the whole swarm population. In the Passive-Scenario unlike Active-Scenario fake peers don't communicate with other peers and they just give maximum score to one another; however, they can't assign score to honest peers because unilateral scores in our system are ignored. In this scenario the correspondent equation for occupying all r Super-Peer positions by fake peers is:

$$\begin{array}{c} r^{th} fake\ peer\ global\ score > Max(honest\ peer\ global\ score) \\ (F-1) > K_g{*}(N-F-1){*}S_p \end{array} \quad (7)$$

In Eq. 7 the second part of fake peer global score and the first part honest peer global score are omitted due to the fact that fake peers don't communicate with other peers and global score of all fake peers is same and equal to *F-1*. Replacing K_g and S_p from Apendix.A parameters shows that fake peers must consist at least %34 to %44 of whole swarm population.

Therefore in both Active and Passive scenarios, fake peers must consist a large amount of swarm in order to occupy all the Super-Peer positions. In addition, as the evaluation results indicate, even when fake peers consist 50 % of swam, they can't occupy all the Super-Peer positions. This is due to the fact that, this analysis is the optimistic case, and in the implementation from the begging, the algorithm detects fake peers gradually, and they can't affect the swarm any more.

4.3 Modified choking algorithm

In this section the modified choking algorithm which is used by peers for Unchoking and Optimistic Unchoking is discussed. The input of this algorithm is a list of peers sorted based on the original BitTorrent tit-for-tat mechanism. This sorting based on the tit-for-tat ensures fairness among peers. After sorting, candidate peers are selected according to their local and global scores in three stages. Algorithm 1. shows the modified choking algorithm.

In the first part of this algorithm, lines 4 through 10, judgment about peers is done using their global and local scores; if their scores are upper than a threshold then they will be unchoked.

```
Algorithm  Rechoke_OptimesticUnchoke()
Begin
1:  maxUnchoked = const;
2:  numUnchoked = 0;
3:  peerList ← Peers' list sorted based on BitTorrent tit-for-tat
4:  foreach(peer(p) IN peerlist WHILE numUnchoked < maxUnchoked)
5:    GS = p.globalScore;
6:    LS = p.localScore;
7:    If(GS ≥ GThresh && LS ≥ LThresh )
8:          peerList.remove(p);
9:          sendUnchoke(p);
10:         numUnchoked ++;
11: foreach(peer(p) IN peerlist WHILE numUnchoked < maxUnchoked)
12:   GS = p.globalScore;
13:   LS = p.localScore;
14:   If(GS ≥ 0 && LS ≥ LThresh )
15:         peerList.remove(p);
16:         sendUnchoke(p);
17:         numUnchoked ++;
18: foreach(peer(p) IN peerlist WHILE numUnchoked < maxUnchoked)
19:   GS = p.globalScore;
20:   LS = p.localScore;
21:   If(GS ≥ 0 && LS ≥ 0)
22:         peerList.remove(p);
23:         sendUnchoke(p);
24:         numUnchoked ++;
25: if(currentReceivedChunk.hash == failded)
26:       SendChoked(chunkSender);
End
```

Algorithm 1 Modified Unchoking algorithm.

In the second part of this algorithm, lines 11 through 17, if there is still unchoke capacity, peers who don't have negative

global score and a local score greater than local score threshold will be unchoked. The third section of this algorithm, lines 18 through 24, tries to prevent performance downfall in the swarm. One of the existing threats in BitTorrent arises when peers don't unchoke one another which decreases performance of swarm significantly. After unchoking peers due to their global and local scores, if there is still capacity to unchoke, peers who have had no suspicious behavior (usually newcomers) will be unchoked. Finally, in the last section of this algorithm, lines 25 and 26, peers which have uploaded fake blocks are choked immediately.

5 Impact evaluation

To evaluate effectiveness of the proposed TMS, we implemented it both in a BitTorrent simulator and in a real BitTorrent swarm in the Emulab [33] testbed. Performance and robustness of the proposed TMS was evaluated against Free-Riders and under different kinds of attacks with various percentages of the rogue peers. Attacks included: 1) Fake-Block 2) Chatty-Peer 3) Lying-Piece and the collusion attack. To clarify differences and characteristics of different kinds of malicious peers, their properties are summarized in Table 1. In the following sections first, simulation results are presented; then, results of the implementation of BitTorrent in Emulab testbed are demonstrated.

5.1 Simulation

We used the JAVA based discrete event General P2P Simulator (GPS) [34] to implement our simulations. GPS is between Message-Level and Packet-Level network simulator. It provides a selection of flow models including: Peer Based Bandwidth (PBB), Dynamic Link-level Network Bandwidth (DLNB) and TCP Based Flow (TBF) Model. Although packet level network simulation is more accurate than message level, but it requires long simulation time. In addition, approximate models ought to be sufficient especially when macro-level phenomena like download amount or time are important. The GPS has a built-in BitTorrent implementation, and this built-in implementation was modified to implement the proposed algorithm.

Table 1 Properties of different malicious peers

	Send Bogus Chunk	Piece-Lying	Fake Trust info	Upload Valid Chunk
Fake-Block Peer	YES	YES	YES	NO
Chatty Peer	NO	YES	YES	NO
Free-Rider	NO	NO	NO	NO

Table 2 Simulation parameters

Property	Value
File size	500 MB
Peers' Bandwidth	1 Mb Down/Up
Chunk size	256 KB
Max number of Unchoked Connections	5
Rechoking interval	10 sec
Number of random peers in Tracker's response	50
Tracker update interval	60 sec
Local Score Threshold	0.2
Global Score Threshold	0.5
request_time_out	3 times of an answered request
rehabiliation_interval	10 min

Each downloading session consisted of 100 peers including one tracker and one peer as the seed. The network topology was generated using GT-ITM [35] which is integrated into GPS. The calculation of local and global trust score thresholds is discussed in the Appendix B. Other simulation parameters are illustrated in Table 2. Each simulation was done to download completion of all peers; however, after spending 60 min, the simulations were terminated. The simulations were performed to evaluate the effect of Fake-Block and Chatty-Peer attack separately. In the proceeding sections first Fake-Block attack results are presented, and then Chatty-Peer attack outcomes are discussed.

5.1.1 Evaluation of fake-block attack

This section illustrates the impact of Fake-Block attack on the performance of honest peers. In order to evaluate the impacts of this attack, honest peers average download speed with the purposed TMS was compared to the original BitTorrent peers. In addition, to measure effectiveness of the proposed TMS in comparison with the previous work [22], we implemented their TMS as well.

Figure 2 shows the average download speed of honest peers with the proposed TMS compared to the original BitTorrent peers and peers with the TMS [22]. The simulations were performed under presence of various percentages of Fake-Block peers with collusion between them. The results indicate a considerable improvement in the download speed of honest peers in comparison with the original BitTorrent and TMS [22] peers. As it is evident in Fig. 2 when more than 30 % of the peers send fake blocks, the average download speed of the original BitTorrent peers decreases significantly. This significant decline is due to the fact that original BitTorrent peers lack any specific mechanism to detect and limit this kind of peers.

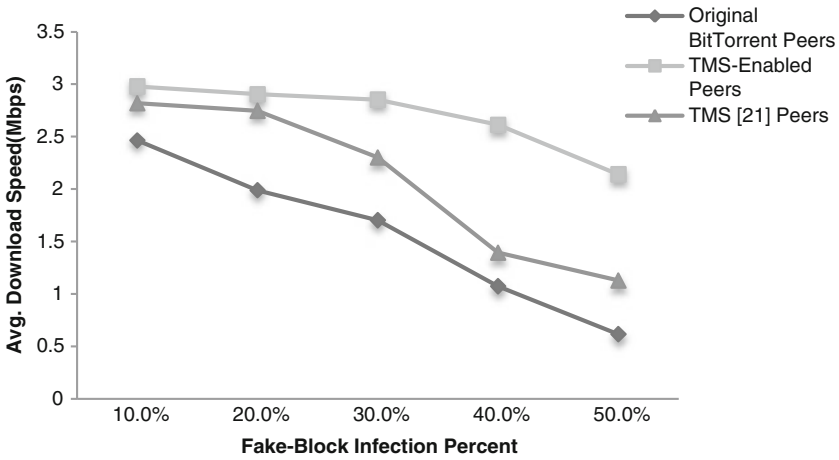

Fig. 2 Average download speed of honest peers with the proposed TMS compared to the original BitTorrent peers and peers with TMS [22]

Moreover, although peers with the TMS [22] performs satisfactory in the presence of 10 % and 20 % Fake-Block peers, their average download speed plunges sharply after presence of more than 30 % Fake-Block peers. This significant decline is obvious in the presence of 40 % Fake-Block peers in which the average download speed of honest peers is approximately similar to original BitTorrent peers. We believe that this sharp decline is result of using a random mechanism at tracker to select peers for judging about other peers global scores. When number of fake peers grows, the chance of a fake peer to be opted as a Super-Peer becomes more and more. In this way false trust information will be propagated, and this adverse downfall in the performance of honest peers will be imminent.

Finally, as it was surmised in the Robustness Analyses section, after growing of fake peers to more than 40 % they can enter the Super-Peer positions. This phenomenon triggers the downfall in the speed of TMS-Enabled peers. However, their speed is still substantially higher than BitTorrent and TMS [22] peers.

5.1.2 Evaluation of chatty-peer attack

In this section evaluation of the proposed TMS facing with Chatty-Peers is presented. Figure 3 shows the average download speed of honest peers with the proposed TMS compared to the original BitTorrent peers and peers with the TMS [22]. Similar to the results of previous section, the average speed of TMS-Enabled peers is substantially higher than the original BitTorrent and TMS [22] peers. Moreover, also in Chatty-Peer attack, the average download speed of peers with TMS [22] declines significantly in the presence of more than %30 chatty peers. This decline as well as the decline in the Fake-Block attack indicates that using a random mechanism at the tracker to find Super-Peers which determine the global score of other peers is not reliable.

Another considerable point about Fig. 3 is that the impact of Chatty-Peer attack on the download speed of honest peers is less effective than Fake-Block attack and shows that Fake-Block attacks are nearly 15 % more harmful and effective than Chatty-Peer attacks. One reason can be that honest peers spend a considerable time for detection of fake blocks because they first should download the whole block and then identify the corruption of that block.

5.2 TMS implementation results

In order to evaluate the proposed TMS in a real BitTorrent swarm we used Snark [36], an open-source implementation of BitTorrent in Java. Snark was manipulated to include the proposed TMS.

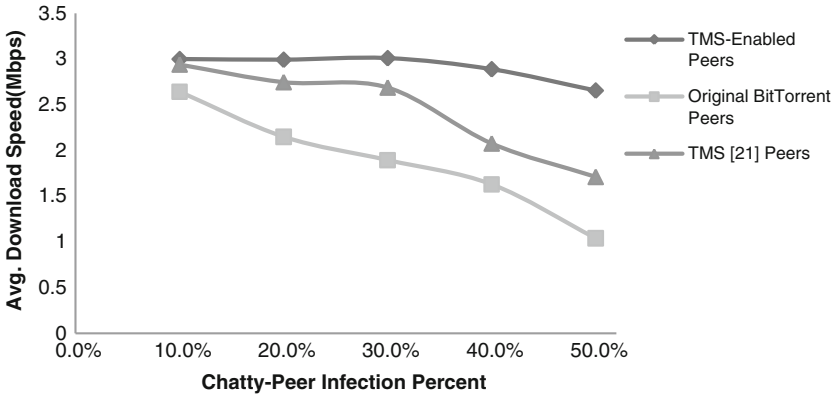

Fig. 3 Honest peers average download speed with the proposed TMS compared to original BitTorrent and TMS [22] peers in presence of various percents of Chatty-Peers

Table 3 Number of different node types

Node type	No. of nodes first exp.	No. of nodes second exp.
TMS-Enabled Honest Peers	36	26
Tracker	1	1
BitThief Client	1	1
Vuze Client	1	1
Transmission Client	1	1
Chatty Peers	5	10
Fake-Block Peers	5	10
Total	50	50

The Emulab testbed was used to perform the real experiments. A 200 MB file was shared in two distinct swarms with different number of malicious peers. In the first swarm malicious peers consisted 20 % of swarm, and in the second swarm they consisted 40 % of the whole swarm population. In each swarm there were totally 50 nodes of six different types which are summarized in Table 3. All other properties except the bandwidth were same as the simulator configurations in Table 2. In order to ensure that low-bandwidth peers would not be penalized, different bandwidth was assigned to various peers. To achieve this goal, each peer's bandwidth was selected randomly from the set of 256 Kbps, 512 Kbps, and 1024 Kbps. Moreover, to spread peers in different clusters we developed a random graph generator which outputs the Tcl correspondent code. The source codes for our BitTorrent Implementation with TMS and Emulab Topology generator are publicly available at: http://sbu-alumni.ir/behrooz/projects.html. Figure 4 is an example of generated graph and nodes distribution with automatic graph generator.

As Fig. 4 indicates, a combination of different types of nodes was experimented. In addition, to assess the effectiveness of the proposed TMS in prevention of free riding, BitThief, the prominent BitTorrent Free-Rider was included in the swarm as well. Each node was run to completion of its download; however, after spending 100 min all nodes were terminated. In order to compare different nodes performance, the ratio of each node average speed to its bandwidth was calculated and demonstrated in Fig. 5.

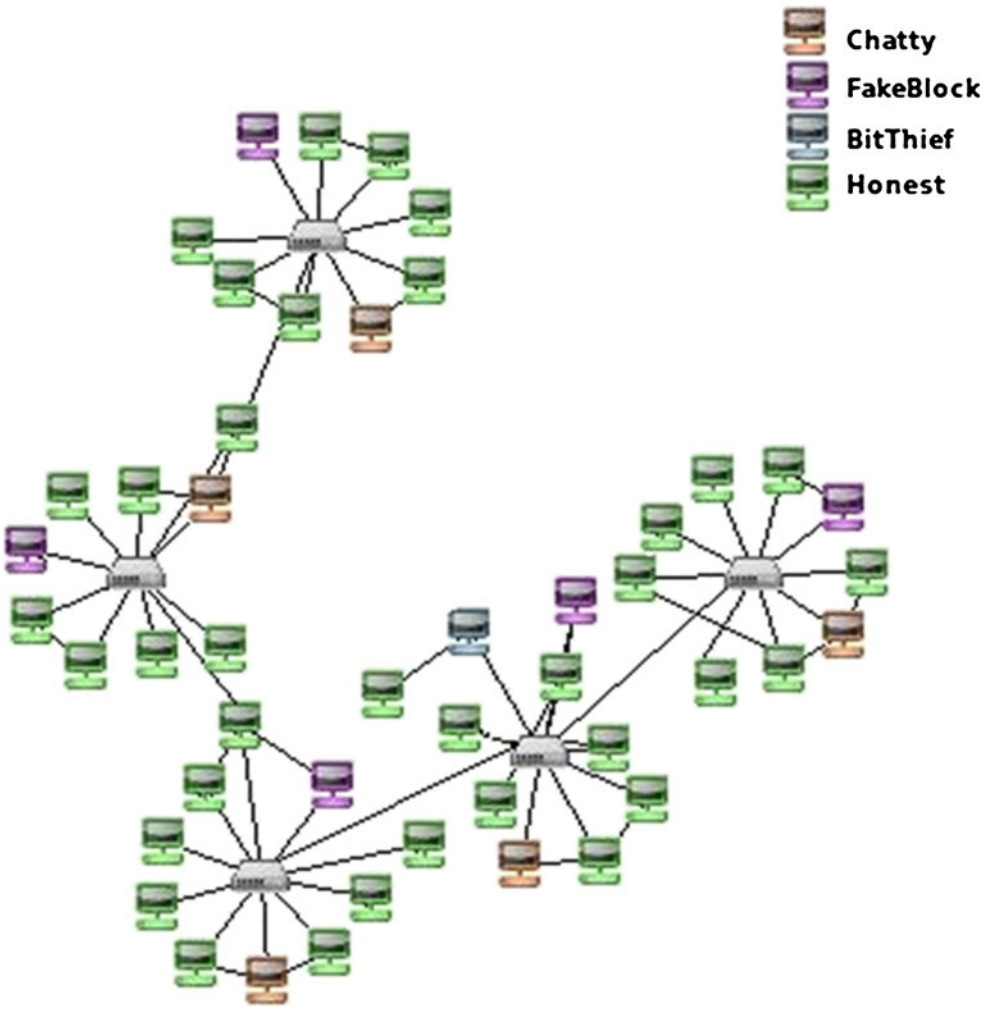

Fig. 4 An example of generated graph with the automatic random graph generator for the Emulab testbed

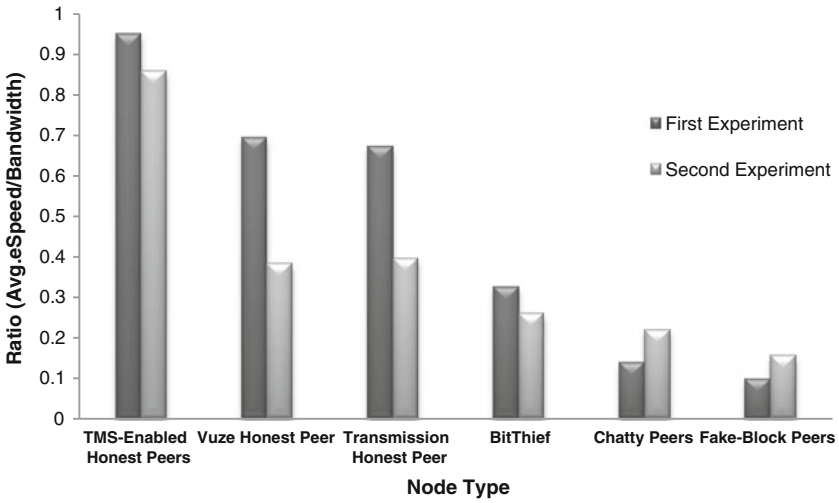

Fig. 5 The ratio of each node type average speed to its bandwidth

As Fig. 5 demonstrates, the TMS-Enabled peers have a considerable higher performance in comparison with other peers in the both experiments. In addition, two of famous BitTorrent Clients, Vuze and Transmission were experimented as well. As the results show, in the first experiment the TMS-Enabled peers were nearly %35, and in the second experiment more than twice more successful than Vuze and Transmission clients. This increase is due to the fact that the ordinary clients are adversely affected by Chatty and Fake-Block peers activities because they don't utilize the trust information at the tracker. In the first experiment the number of malicious peers is half of the second experiment; consequently, the Vuze and Transmission reach a higher ratio, but in the second experiments, as the number of rogue peers increase their performance decreases. Therefore, it is evident that the ordinary BitTorrent clients are vulnerable to misbehavior in the swam. In contrast with Vuze and Transmission clients, TMS-Enabled peers, based on their knowledge of other peers can easily distinguish these activities, and avoid communication with malicious peers.

In addition to malicious activities such as sending fake blocks (Fake-Block attack) or evading from responding (Chatty attack), Free-Riding is substantially penalized in the proposed TMS. According to Fig. 5, the performance of TMS-Enabled peers is approximately three times higher than BitThief, one of the most famous Free-Riding clients.

Finally, the significant low performance of Chatty and Fake-Block peers even in the second experiment demonstrate that their behavior is recognized during early stages of download process. In addition, they will be choked and cannot delay TMS-Enabled peers any longer. The main reason for the low download speed of Chatty and Fake-Block peers is that the number of TMS-Enabled peers is high and they detect the malicious peers, and choke them; as a result, malicious peers speed declines significantly.

5.3 Proposed TMS robustness evaluation

The main threat for the proposed TMS is collusion attack and entrance of rogue peers in Super-Peers set which can result in significant decrease in the performance of swarm. On the other hand, we believe that through the repetition of Super-Peer selection process at the tracker, number of rogue peers in Super-Peers set approaches a small constant. To show this fact, number of rogue peers which could enter in the Super-Peers set was measured, in the presence of 30 %, 40 %, and 50 % rogue peers. This measurement was performed in the GPS simulator with total number of 100 peers during 15 min.

As Fig. 6.a, b and c indicate, after some overshoot in the beginning of downloading process, number of rogue peers selected as Super-Peer, approaches a constant value, like a damping wave. At the beginning of Super-Peer selection process, peers behaviors are still unknown; therefore, the TMS needs an amount of time to pass the transit state and reaches to a steady state. In our simulation results, this required transit time was nearly 11 min, but this time can vary in different conditions, and depends on different parameters such as type and entrance rate of rogue peers. Usually this time compared to life time of a swarm is inconsiderable and TMS can reach a steady state during early phases of its life. Thus, after passing the transit state, the proposed TMS can prevent rogue peers to be selected as Super-Peer.

6 Tms imposed overhead

In this section imposed overhead on BitTorrent clients and the tracker by the proposed TMS is discussed. In the proposed TMS, list of peers transactions and correspondent scores are kept at the clients; as a result, transactions history is distributed along the peers. This distribution of transactions causes a negligible overhead on each peer individually and each peer just keeps track of its transactions and computes score for its transactions.

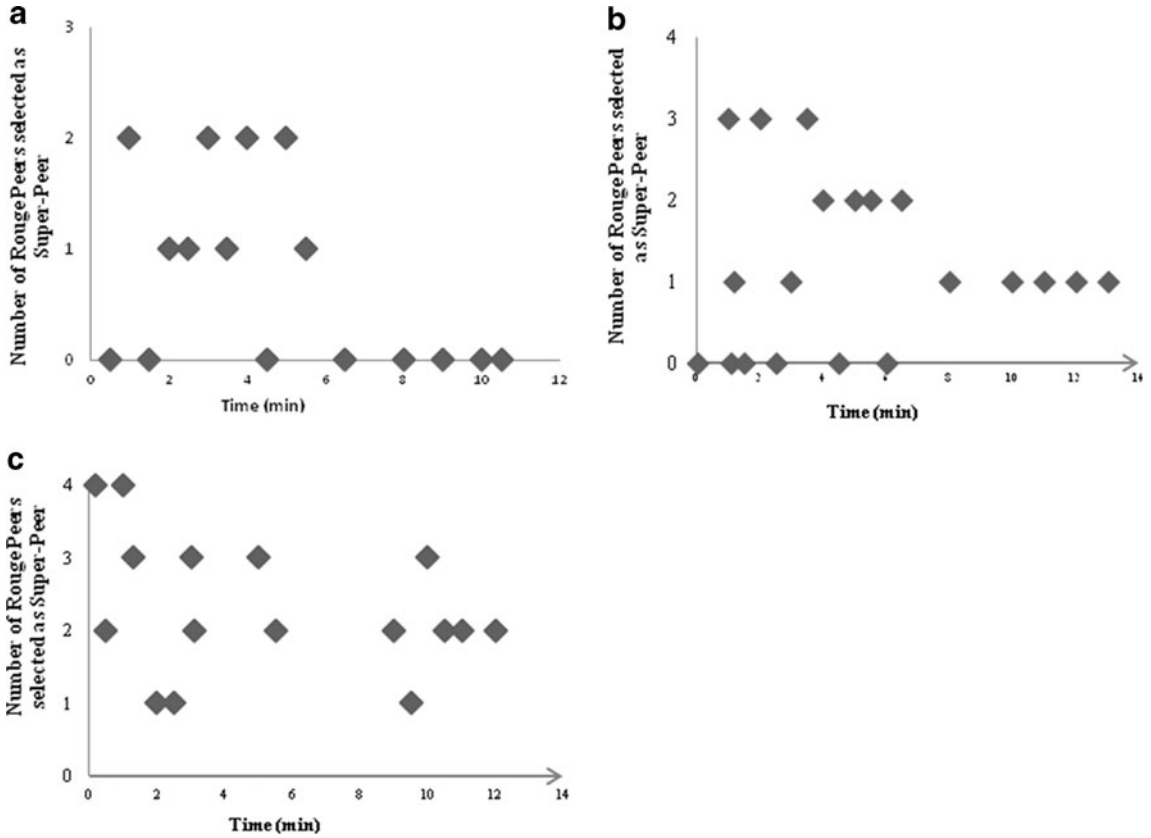

Fig. 6 **a** Number of Rogue Peers selected as Super-Peer when. **b** Number of Rogue Peers selected as Super-Peer when 30 % of swarm is infected. **c** Number of Rogue Peers selected as Super-Peer when 50 % of swarm is infected. 30 % of swarm is infected

The imposed overhead on the tracker is important as well because today's trackers are required to do a small amount of work. The only additional duty of tracker is to collect scores from different peers and calculate cumulative score of each peer when they connect to tracker to announce their status. Therefore, the amount of storage and computation at tracker side is not significant; conversely, the amount of communication on the part of tracker needs a more accurate attention. As it was stated in section 3.1, peers interact with tracker at fixed intervals and they send their progress and local scores at these intervals as well. Thus, no extra connection in comparison with the original protocol, to the tracker is initiated and only the amount of data exchanged during each interval between peers and tracker is slightly increased.

In summary, the conclusion can be drawn that, the proposed TMS does not imposes a significant overhead neither on peers nor on the tracker.

7 Future works

One of the considerable topics about the proposed TMS is the tracker, which can become a single point of failure and susceptible to a DoS attack. In addition, relying on the tracker to calculate the global trust scores, adds overhead to the tracker when the size of the swarm is extremely large. As it was stated in the section 6 there is no connection overhead but the amount of storage and calculation overhead in extremely large swarms may become a bottleneck and significant. The Torrentfreak website [37] reports that the largest BitTorrent swarm (seeds and leechers) ever witnessed was an episode of a serried entitled "Heroes" with a total of 144,663 peers. In such large and crowded swarms the overall imposed overhead on the tracker can become significant because even these highly small amount of extra storage and calculation per peer when observed as a whole become huge amounts.

In order to solve these issues, BitTorrent has drafted an addition to its protocol that allows for trackerless torrents. In the trackerless mode a distributed hash table (DHT) is used instead. Usually most of client implementations, support Peer Exchange (PEX) which allows peers to supplement the trackers functionality by sharing peers lists among themselves. In an attempt to address both of expressed concerns, we are designing and evaluating a distributed tracker protocol for our TMS using DHT.

8 Conclusion

In this paper, we have presented a new reputation based trust management system for BitTorrent protocol. The goal of our work is to cover different attack types against BitTorrent without changing the present protocol a lot. The main concept of our TMS is sharing experiences among peers. We used both local and global score concepts to detect malicious peers as fast as possible. The tracker's functionality is extended to collect local scores as well as calculation and dissemination of global scores.

The results of simulation and implementations indicate that the proposed TMS can improve performance of honest peers significantly. In addition, our TMS was highly successful in limiting activity of rogue peers and minimizing downloading by Free-Riders. Another considerable point about the proposed TMS is its robustness in presence of large amount of rogue peers with collusion among them. Our analyses and experiment results show that even in presence of 50 % percent rogue peers in swarm the proposed TMS can act and decide rationally.

Acknowledgments This research has been done by financial support of Shahid Beheshti University research chancellor under Contract No.: 600/537-90/3/30.

Appendix A *Calculation of* S_n, S_p, K_g *and* K_f *parameters*

In order to estimate S_n parameter we collected a huge amount of data samples. By plotting data and fitting probability curves, we discovered that fake peers' scores follow a normal distribution. As a result, in order to estimate the average score of fake peers given by honest peers, statistical confidence interval estimation was used. A %85 confidence interval for the average score of fake peers was calculated as following:

$$\text{A } 100*(1-\alpha)\text{ confidence interval for average of data}(\mu):$$
$$\mu \in (\bar{x} - z_{1-\frac{\alpha}{2}}(n-1)\frac{s}{\sqrt{n}}, \bar{x} + z_{1-\frac{\alpha}{2}}(n-1)\frac{s}{\sqrt{n}})$$

Where μ is average score of fake peers, $\bar{x}$ and s are standard average and deviation of samples, n is number of samples and (1-α) equals 0.85. Replacing these values, we have μ ∈ (−0.9223, −0.7023) confidence interval for average score of fake peers. By a similar analysis to S_n, the following intervals for other parameters were estimated.

Parameter	Confidence-Interval
S_p	(+0.6703,+0.8803)
K_g	(+0.6326,+0.8504)
K_f	(+0.7806,+0.8308)

Appendix B *Calculation of* **local** *and* **global** *trust score thresholds*

In order to find optimal values for the local and global trust score thresholds, we ran several simulations of a swarm with 100 peers including 25 malicious peers of different types. These thresholds can have a value in the range of [-1, 1]; however, we know that if peer A wants to unchoke peer B, peer B should have at least a positive score. Therefore, we limited the possible range for these parameters to [0, 1]. In addition, as it was discussed in the section 4.3 the excessively high values for these parameters can cause performance downfall in the swarm; consequently, the range was limited to [0, 0.7]. All the simulations parameters except the local and global trust score thresholds were same as those in the section 5. Figure 7.a and b show the results of simulation for these parameters. As these figures indicate, 0.2 is the best value for the local trust score threshold and 0.5 is the best for the global trust score threshold.

Moreover, in order to evaluate the effect of different values of local and global thresholds on the robustness of proposed system and the number of fake peers selected as super-peer, we used some of the combinations of these thresholds. We

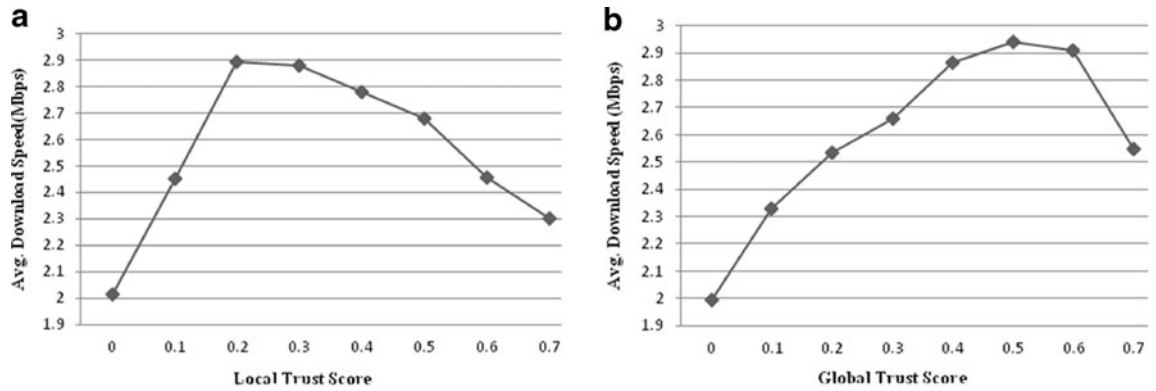

Fig. 7 **a** the average download speed of honest peers with different values for the local trust score thresholds and same global trust score thresholds. **b** the average download speed of honest peers with different values for the global trust score thresholds and same local trust score threshold

tested six different combinations of 0.2, 0.3 for local score threshold and 0.4, 0.5, and 0.6 for global trust score threshold. The simulations were performed in GPS simulator in a swarm of 100 peers with 40 malicious peers of different types. The following table shows the number of malicious peers selected as the super peer after 15 min during which system reaches a steady state. As the results indicate, the proposed mechanism is not highly dependent on these thresholds and all the values for these thresholds within the given ranges have a similar effect on the robustness of system.

Local & Global Trust Score	Number of malicious peers selected as super-peer
Local: 0.2 Global:0.4	1
Local: 0.2 Global:0.5	2
Local: 0.2 Global:0.6	1
Local: 0.3 Global:0.4	1
Local: 0.3 Global:0.5	1
Local: 0.3 Global:0.6	1

References

1. BitTorrent. http://www.bittorrent.com/. Accessed Dec. 2011
2. Ipoque, "Internet study 2007: Data about P2P, VoIP, Skype, file hosters like Rapidshare and streaming services like YouTube". http://www.ipoque.com/resources/internet-studies/internet-study-2007/,2007. Accessed Dec. 2011
3. Banerjee A, Faloutsos M, Bhuyan L (2007) "Is someone tracking P2P users?". Proc IFIP NETWORKING, Atlanta, GA
4. BitTorrent servers under attack. http://en.wikipedia.org/wiki/Torrent_poisoning/. Accessed Dec. 2011
5. Ziptorrent blacklist. http://torrentfreak.com/ziptorrent-pollutes-and-slows-down-popular-torrents/. Accessed Dec. 2011
6. Kong J, Cai W, Wang L, Zhao Q (2010) "A study of pollution on BitTorrent", Proc. The 2nd International Conference on Computer and Automation Engineering, Singapore, Feb. 2010
7. Konrath MA, Barcellos MP, Mansilha RB (2007)" Attacking a swarm with a band of liars: evaluating the impact of attacks on bittorrent", Proc. IEEE P2P, Galway, Ireland
8. Shin K, Reeves DS, Rhee I (2009) "Treat-before-trick: Free-riding prevention for BitTorrent-like peer-to-peer networks", Proc. IEEE International Symposium on Parallel&Distributed, pp 1–12
9. Locher T, Moor P, Schmid S, Wattenhofer R (2006) "Free riding in bittorrent is cheap", Proc Hot-Nets
10. Piatek M, Isdal T, Anderson T, Krishnamurthy A, Venkataramani A (2007) "Do incentives build robustness in bittorrent?", Proc. 4th USENIX Symposium on Networked Systems Design & Implementation, pp 1–14
11. Maini S (2006) "A Survey Study on Reputation-Based Trust Management in P2P Networks", Technical Report, Department of Computer Science; Kent State University
12. Cohen B (2003) "Incentives build robustness in BitTorrent", Proc. 1st Workshop on Economics of Peer-to-Peer Systems
13. Levin D, LaCurts K, Spring N, Bhattacharjee B (2008) "BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives", Proc. SIGCOMM'08, Seattle, Washington, USA
14. Blaze M, Feigenbaum J (1996) "Decentralized Trust Management", Proc. IEEE Symposium on Security and Privacy
15. Kagal L, Cost S (2001) "A framework for distributed trust Management", Proc. Second Workshop on Norms and Institutions in MAS, Autonomous Agents
16. Sabater J, Sierra C (2002) "Reputation and social network analysis in multi-agent systems", Proc. First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy
17. Pujol J, Sanguesa R (2002) "Extracting reputation in multi agent systems by means of social network topology", Proc. First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy
18. Kamvar SD, Schlosser MT, Garcia-Molina H (2003) "The Eigen Trust algorithm for reputation management in p2p networks". Proc. 12th International World Wide Web Conference
19. Hu J, Li X, Zhou B, Li Y (2010) "SECTrust: A Secure and Effective Distributed P2P Trust Model", Proc. IITSI, pp 34–38
20. Stakhanova N, Ferrero S, Wong J, Cai Y (2004) "A reputation-based trust management in peer-to-peer network systems," Proc. International Workshop on Security in Parallel and Distributed Systems, San Francisco, CA
21. Singh A, Liu L (2004) "TrustMe: Anonymouss management of trust relationships in decentralized P2P systems", Proc. The Third IEEE International Conference on Peer-to-Peer Computing, Linkopings
22. Shah P, Pâris J-F (2007) "Incorporating trust in the BitTorrent protocol", Proc. of SPECTS, Paris
23. Chen H, Ye Z, Liu W, Wang C (2009) "Fuzzy Inference Trust in P2P Network Environment", Proc. International Workshop on Intelligent Systems and Applications, Wuhan, pp 1–4
24. Liu F, Ding Y (2007) "Ecological Network-Inspired Trust Management Model of P2P Networks", Proc. Second Workshop on Digital Media and its Application in Museum & Heritages, pp 297–302
25. Gupta M, Judge P, Ammar MA (2003) "A reputation system for peer-to-peer networks". Proc. NOSSDA V
26. Gheorghe G, Cigno RL, Montresor A "Security and Privacy Issues in P2P Streaming Systems: A Survey", Springer. Peer-to-Peer Networking and Applications, 2010, Springger New York, pp 1–17
27. Douceur JR (2002) "The Sybil attack", Proc. 1st International Workshop on Peer-to-Peer Systems, Cambridge, MA, USA, pp 251–260
28. Dhungel P, Wu D, Schonhorst B, Ross KW (2008) "A measurement study of attacks on BitTorrent leechers", Proc. IPTPS, Tampa Bay, FL, USA
29. Conner W, Nahrstedt K, Gupta I (2006) "Preventing DoS attacks in peer-to-peer media streaming systems", Proc. 13th Annual Conference on Multimedia Computing and Networking (MMCN'06), San Jose, CA, USA
30. Yang J, Li Y, Huang B, Ming J (2008) "Preventing DDoS attacks based on credit model for P2P streaming system", Proc. 5th international conference on Autonomic and Trusted Computing (ATC'08), Berlin, Heidelberg, pp 13–20
31. Kohno T, Broido A, Claffy K (2005) Remote physical device fingerprinting. IEEE Trans Dependable Secure Comput 2(2):93–108
32. Bazzi R, Konjevod G (2005) "On the establishment of distinct identities in overlay networks", Proc. ACM Symposium on Principles of Distributed Computing, Las Vegas, NV
33. Emulab - Network Emulation Testbed. www.emulab.net. Accessed Aug. 2011
34. Yang W, Abu-Ghazaleh N (2005) "GPS: a general Peer-to-Peer simulator and its use for modeling BT", Proc. 13 Int. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Atlanta, GA
35. Zegura EW, Calvert KL, Bhattacharjee S (1996) How to model an internetwork. Proc IEEE Infocom USA 2:594–602
36. The Hunting of the Snark Project, http://klomp.org/snark/, Accessed Oct. 2010
37. Indie Band Tops a Million Downloads, Breaks BitTorrent Record. http://torrentfreak.com/indie-band-tops-a-million-downloads-breaks-bittorrent-record-110317/. Accessed Aug. 2011

Behrooz Shafiee Sarjaz is an undergraduate student in computer engineering at Shahid Beheshti University. He is currently working on network security with focus of P2P networks security. His main research interests include Network Security, Distributed.

Maghsoud Abbaspour received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 1992, 1995 and 2003, respectively. He has been an assistant professor of Computer Engineering department and the manager of Wireless Networking Laboratory in Shahid Beheshti University, Tehran, Iran since 2005. He is interested in Wireless Sensor Networks, peer to peer and ad hoc networks, network security and multimedia networking.