Incorporating Trust in the BitTorrent Protocol

Purvi Shah **Jehan-François Pâris**
Department of Computer Science
University of Houston
Houston, TX 77204-3010
{purvi, paris}@cs.uh.edu

Keywords: Peer-to-Peer network, BitTorrent protocol, trust management system, fairness

Abstract

We present a trust management system tailored to the needs of the BitTorrent protocol. Unlike other trust management systems, our system is partially decentralized and assumes unstructured BitTorrent swarms. In particular, we use the tracker of each BitTorrent swarm, to compute and distribute global trust scores complementing the local trust scores maintained by each peer. Our simulation study indicates that our system significantly minimizes the downloading by a rogue peer in the swarm. We also found out that it resulted in a fairer allocation of chunks among peers with different link bandwidths.

1. INTRODUCTION

In recent years, Peer-to-Peer (P2P) technology has captured the interest of both the industry and the research community. By allowing peers to serve each other, P2P solutions overcome many limitations of traditional client server architecture. They can handle very large and sudden surges of demand (flash crowds) as well as overcome server bandwidth limitations. In addition, P2P solutions do not require any special support from the network, let it be IP multicast or any specific content distribution infrastructure.

Applications using P2P architectures can now be found in such diverse domains as the medical field, in entertainment, in education, in software management and updates, and so on. P2P communications now account for 40 to 80 percent of the Internet backbone traffic [Par04]. Yet the understanding of these new applications is incipient.

Underneath the potential benefits of P2P networks lays the challenge of providing both robustness and performance. One of the main issues to be considered is how to protect P2P architectures against malicious attacks, selfish behaviors and software errors. A number of exploits on P2P networks are possible including Distributed Denial of Service (DDoS) [WP02] where some peers may spoof and deceive others. Such *rogue* peers may exploit their identity and clog up network links, thereby preventing legitimate network traffic.

Trust management systems (TMSs) [RZ+00] were introduced to solve these problems. They reward peers that cooperate with other peers, punish rogue peers and finally motivate or even force the peers to cooperate with each other. TMS assign to each peer a *trust score* that can be used by other peers to decide whether or not to interact with that peer.

Most TMSs [KSG03, DV+02, XL04, SF+04 and ZH06] have focused on fully decentralized P2P networks such as *Gnutella* [Gnu07]. However in practice partially decentralized P2P networks (for instance, P2P networks using the *BitTorrent* [Coh03] and *FastTrack* [Fas07] protocols) are widely used and make up more than 70 percentage of the P2P traffic on the Internet [Par04]. Each specific P2P network brings about its own set of problems based on its protocol and application area and therefore TMS for fully decentralized P2P networks are poorly suited to partially decentralized networks. Only *Kazaa* [Kaz07] P2P network (using the FastTrack protocol) makes use of *participation level*, somewhat like a trust score for each peer based on the amount of data it exchanges and the integrity of the content it uploads.

In this paper we are primarily concerned in the performance of P2P networks using the BitTorrent (BT) protocol. BT is a very scalable second generation P2P content distribution protocol that we describe in detail in the next section. BT-based file distribution has been adopted by several Internet content providers [Wag07, Bow07]. Current implementations of the BT protocol let peers verify that the chunks they download from other peers were not tampered with. However they do not prevent rogue peers to slow down the data distribution process by flooding other peers with bogus chunks or by acting selfishly. A wider acceptance of the BT technology requires the implementation of effective measures against these possible attacks.

The rest of the paper is structured as follows: Section 2 presents a quick overview of BT. Section 3 explains the vulnerabilities in BT *swarms*. In section 4 we discuss related work. Section 5, 6 and 7 presents our TMS for the BT protocol to enhance its performance. In section 8 we describe our experimental setup and present our results. Finally Section 9 has our conclusions.

2. BITTORRENT OVERVIEW

Unlike more traditional P2P protocols such as Gnutella and FastTrack, BT is designed for bulk data distribution and

works by chopping up the file and delivering the chunks in a non-linear format.

BT differentiates between two types of peers: *leeches* and *seeds*. Leeches are peers that only have some or none of the data while seeds are peers that have all the data but reside in the network to let other peers download from them. Thus seeds only perform uploading while leeches download chunks that they do not have and upload chunks that they have.

BT implements a set of algorithms that balances the load of content distribution among a *swarm* constituting an overlay network of peers. A centralized process, the *tracker*, manages each swarm. This tracker does not host any content but maintains metadata about it. As peers enter the swarm they first connect to the tracker. The tracker returns a random list of peers that have the content. Each peer then randomly selects a subset of that list as its *neighbors* and initiates requests to set up bi-directional TCP connections with these neighbors.

To amplify the overall efficiency of the swarm, BT employs a *rarest first* policy for selecting chunks to download. Peers attempt to download a chunk that is least replicated among its neighbors. This way the peers share chunks with one another, instead of downloading directly from the source-server.

Peers optimize their download and upload rates by means of the *tit-for-tat* policy. Peers typically upload to the k peers that recently provided it with the best downloading rate, even though it may have received requests from more than k of them. This process of temporary refusal to upload to some peers is called *choking*. It lets each peer attempt to maximize its own interest by downloading as much as it can.

BT also includes a mechanism, called *optimistic unchoking*, that lets peers reserve a share of their available bandwidth for uploading chunks to randomly selected peers. Among other things, optimistic unchoking gives new coming peers a chance to join the swarm. The decision to choke/unchoke is performed at regular *rechoking* intervals.

3. VULNERABILITIES IN BT SWARMS

As BT swarms continue to grow in popularity the threat of rogue peers participating in large swarms will also increase. DDoS attacks are common online assaults that aim to overwhelm network bandwidth using a flood of data, to prevent access to servers. Unfortunately, BT protocols do little to protect against such attacks.

Another critical problem with P2P networks is that of *free riding* (that is, when peers behave selfishly and download chunks for themselves without contributing to the swarm). While recent studies [BHP06, JA05 and TC05] have shown that the BT incentive mechanism appears fairly robust against rogue peers, they have also found out that it can sometime induce free riding because it does not effectively reward good peers and punish rogue peers.

BT's incentive mechanism is *rate-based*, which can create unfairness in terms of the amount of chunks exchanged. Rogue peers can obtain more bandwidth and honest peers download rates may suffer consequently. Bharambe *et al.* [BHP06] have proposed a quick bandwidth estimation mechanism through some probing techniques. Unfortunately, their mechanism is rarely applicable because reliable bandwidth estimation is difficult to achieve in real networks. Another mechanism to alleviate the unfairness is by using an additional condition, *fairness score* that represents the amount of unfairness on P2P connections. Previous studies use this mechanism in different manner: some enforce tit-for-tat at the chunk level [BHP06 and JA05] and others at the byte level [TC05]. Our work continues this approach.

Other studies [LM+06, LN+06 and SP+07] have identified exploits that potentially deliver increased benefits for rogue peers. For instance rogue peers are able to exploit the opportunistic unchoking mechanism. By maintaining a larger list of neighbors, rogue peers may be able to increase the number of optimistic unchokes. An even worse scenario is when a rogue peer connects only to seeds. When such rogue peers download from the seeds, they use up the upload bandwidth of the seeds that could have been used by the honest peers.

4. RELATED WORK

Aberer and Despodovič presented the first TMS specifically designed for P2P networks [AD01]. Methods portrayed in [MG04, ZL04] utilize only local computations on sub-graphs of the whole P2P network.

Eigentrust [KSG03] maintains a global trust score for each peer i computed from the trust score of i given by all other peers weighted by their own global trust scores. However the technique does not address the issue of personalization. In addition, it does not take into account the possibility of *collusion attacks*, where rogue peers collude among themselves peers in order to malign the reputation of honest peers.

P2P Rep [DV+02] bases trust computation on the number of downloads which leaves out peers only sharing content on the network. *Credence* [WS05] proposes object reputation system that addresses the problem of content pollution. *PeerTrust* [XL04] maps a small database that stores a portion of the global trust scores and requires cooperation from the peers for storing the trust score.

Stakhanova *et al.* [SF+04] proposed a TMS that lets each peer to compute individual trust scores rather a global trust score for each peer in the network. Unfortunately, TMS that do not utilize the global trust scores have slower *convergence* and cannot react as rapidly as they should to changes in peer behavior. Incentive systems for fighting *free riding* in P2P networks are proposed in [FL+04].

Most proposed TMSs [KSG03, DV+02, XL04 and ZH06] are based on some form of DHT overlay network

even though most P2P networks deployed are unstructured. These TMSs are designed for fully decentralized P2P networks and use a reactive approach that requires cooperation from a large number of peers for computing the trust scores. This causes additional overhead and latency.

Our approach is tailored to the needs of partially decentralized P2P networks and is based on unstructured BT swarms. Since BT is so dependent on the upload and download rates it does not seem right to only use trust sores. Hence in this work unlike preceding studies we use the existing incentive system in BT (that is, the tit-for-tat policy) in conjunction with our TMS. The trust scores of the neighbors form the basis of the incentive system and are used to guide peers in their decision-making.

5. TRUST MANAGEMENT SYSTEM FOR BT

Our approach uses trust scores based on each peer's interaction with other peers to identify rogue peers. Its design is motivated from the Debit-Credit Reputation Computation System (DCRCS) previously proposed by Gupta et al. [GJA03].

DCRCS tracks positive peer's contribution to a P2P network using a debit-credit mechanism. Each peer assigns trust scores to the peers with which it interacts and stores these trust scores locally. DCRS also includes a *reputation computation agent* (RCA) that occasionally gathers reputation from peers in order to ensure distributed access to the trust scores. We extend the main philosophy of the DCRCS to capture malicious and selfish behavior in BT.

Our TMS works as follows. Each peer assigns a trust score to each of its neighbors. In addition, we use the tracker of each BT swarm to maintain global trust scores. Recall that the tracker is a peer that assists in the communication between peers using the protocol by keeping complete membership information. As peers enter the swarm they first connect to the tracker. Peers that have already begun downloading also communicate with the tracker at fixed intervals to learn about new peers and provide statistics. We call this interval the *tracker update interval* and make use of the tracker collecting information on the peers in the swarm as a central rendezvous point.

Our smart tracker thus extends the functionality provided by the RCA in DCRCS. At every tracker update interval, each peer sends to the tracker the trust scores of the peers it is connected to. The tracker's reply includes the global trust scores. Observe that our approach does not result in any increase in the network traffic as the trust scores are merely piggybacked to the messages already exchanged between the tracker and the peers.

At every rechoking interval, each peer will use the global trust score and the individual trust score of its neighbors to perform its choking/unchoking routine. The peer picks out other peers with the best downloading rate (uploading rate in case of seeds) but unchokes only the selected k peers only based on additional conditions (explained in section 6.3). The rest of the neighbors are choked.

We should note that the trust scores capture the *behavior* of the peers in the swarm while the incentive mechanism captures their *capability*. The behavior of a peer is determined by the intent of the peer whereas its capability depends on its processing capacity and its upload bandwidth. By selectively preferring trustworthy peers to unchoke we can enhance the overall network speed and improve general performance in the swarm for all peers. Thus our comprehensive solution aims to combat rogue peers that can undermine the utility of the swarm.

6. TRUST SCORES

The trust score computation is based on the interaction of peers with others in the swarm. We now explain how the different trust scores are computed.

6.1 Individual trust

The BT protocol lets peers verify that the chunks they download from other peers were not tampered with. We use this information to detect rogue peers. Additionally to detect selfish behavior each peer maintains a *fairness score* f_{ij} of a P2P connection defined as

$$f_{ij} = u_{ij} - d_{ij}$$

where,

u_{ij} is the number of chunks that i uploaded to j

d_{ij} is the number of chunks that i downloaded from j

Peer i then uses this f_{ij} to set individual trust score IT_{ij} of peer j as follows

$$IT_{ij} = \begin{cases} -1 \text{ (if bogus chunk is uploaded by peer } j) \\ 0 \text{ (if } f_{ij} > \theta) \\ 1 \text{ (if } f_{ij} \leq \theta) \end{cases}$$

where $\theta \geq 1$ is the *fairness threshold*. Negative *fairness threshold* would signify that the peer is downloading more than it is uploading and vice versa. If the fairness threshold is not greater than or equal to 1, fewer connections will endure, as it is difficult to find peers with identical upload capacity.

A peer maintains information about the chunks it exchanges only within a *penalty interval*, which helps with peer rehabilitation.

6.2 Global trust

The global trust score GT_j of a peer j captures the global knowledge on the peer. Every tracker update interval the tracker retrieves the individual trust score for peer j from randomly chosen k peers that are connected to peer j.

By adding redundancy to the computation through having the judgment of several peers used to compute the global trust score for any peer we avoid collusion attacks as well as we are able to evaluate the contribution of the peer to the swarm. We set the value of k, 4 for the simulations. To

support preliminary trust for recently joined peers the initial global trust score are set to the *favorable trust score* which is set 0.75 for the simulations.

The tracker then sets the average of the individual trust scores of the chosen peers as the GT_j. Note that peers do not have a say in the calculation of their own global trust score. This ensures that rogue peers have little or no impact on the computation of their own global trust scores.

6.3 Modification in the choking algorithms

A higher-level description of the modified choking algorithm used during the rechoking and optimistic unchoking mechanisms is as follows

```
noUnchokeConnections = 0
maxUnchokeConnections = k
for each peer j in the peerList while
noUnchokedConnections < maxUnchokedConnections
        if (IT_ij == 1 && GT_j > favorable trust score)    {
            unchoke peer j
            remove peer j from the peerList
            noUnchokeConnections++
        }
for each peer j in the peerList while
noUnchokedConnections < maxUnchokedConnections
        if (IT_ij == 1 && GT_j > 0)      {
             unchoke peer j
             noUnchokeConnections++
        }
```

Each peer first unchokes peers with favorable global trust scores, that is, peers that promote fairness. Thus unlike previous studies [BHP06, JA05 and TC05] our TMS makes it easier for peers to adopt the fairness mechanism. Using global trust scores encourages peers to contribute evenly to other peers in the swarm and help sustain the performance of the swarm.

Our peer list is first sorted during the rechoking mechanism based on the downloading rate (uploading rate in case of seeds). Hence instead of enforcing tit-for-tat at chunk level only as proposed in [BHP06, JA05] our peers take a risk to discover other peers providing higher download rates.

Note that as soon as a bogus chunk is downloaded the connection to the rogue peer is closed.

7. DISCUSSION

The assumption of the tracker being used as a central rendezvous point to collect and disseminate trust scores is a bit restraining; it should be noted that the involvement of tracker in the actual content distribution is minimal and therefore it is never a bottleneck for the normal operation of the swarm. Furthermore, since only the tracker is the pre-trusted entity in the swarm the managerial task of ensuring this behavior presents modest overhead.

Table 1. Simulation Parameters

File size	100 MB
Chunk size	256 KB
Total number of peers	100
Number of initial-seeds/source-servers	1
Maximum number of concurrent upload transfers *(maxUnchokeConnections)*	5
Rechoking interval	10 s
Tracker update interval	60 s
Optimistic unchoking interval	30 s
Number of random peers returned by the tracker	50
Number of neighbors of each peer	10
Number of random peers chosen by the tracker to compute global trust scores	4
Penalty interval	9 min
Favorable trust score	0.75
Fairness threshold (θ)	2

Finally, though the tracker appears to be a single point of failure in practice several distributed trackers can be deployed and mapping techniques can be used to direct peers to different trackers to accomplish load balancing [Bit07b, Tra07]. Thus, if one tracker should stop working, the others can keep on supporting the content distribution. The issues in the design of distributed tracker are beyond the scope of this paper but we do believe our TMS can be extended to support such a distributed tracker BT swarm.

Another assumption we rely on is the persistence of a peer's identity. In practice this can be accomplished by refusing multiple connections from a single IP address. Other possible defense mechanism would be to rely on techniques to identify specific hardware hosts [BK05, KBC05]. These techniques have the advantage of also protecting swarms against Sybil attacks, where one single rogue host pretends to be many [Dou02]. Simultaneously in our TMS maintaining a single identity is more useful for a peer in increasing its trust score as well as its observed upload bandwidth.

8. EVALUATION

8.1 Simulation Setup

In this section we assess the performance of our TMS with respect to its efficacy at detecting the rogue peers and also at improving the overall performance of the swarm for all peers.

Our simulations are implemented using the JAVA based discrete event General P2P Simulator (GPS) [YA05]. GPS models concurrent uploads of the peer under the various algorithms of BT and calculates the download rates from its neighbors.

We modeled the network transmission and queuing delays but assumed that the network propagation delays could be neglected since they are relevant only for small sized control packets while the downloading time is

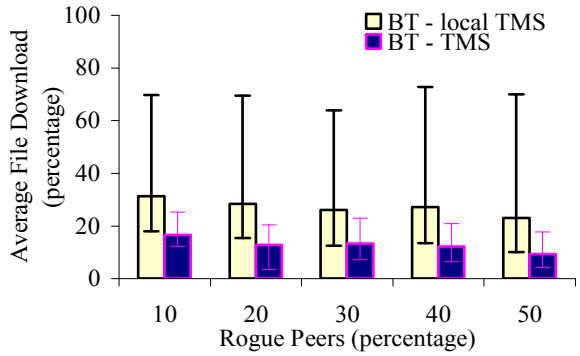


Figure 1. Benefit of TMS using the global trust score

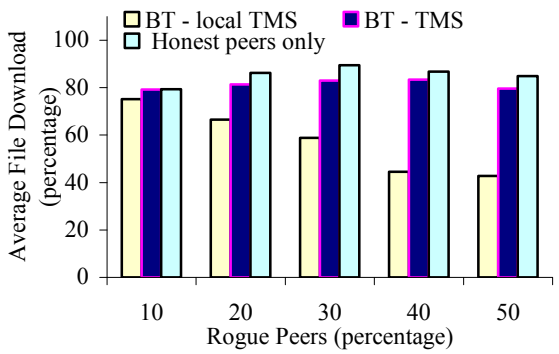


Figure 2. Improvement in the performance of the honest peers in the swarm

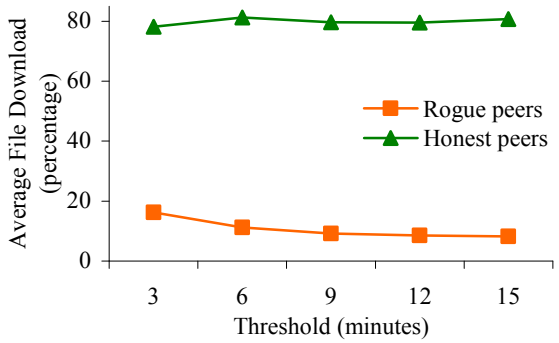


Figure 3. Effect of penalty internal on the performance of the swarms with 50 percent rogue peers

dominated by the chunk exchange traffic. To keep our model simple, we ignored the complexity of the dynamics of TCP connections. We assumed the idealized performance of TCP and assumed that connections traversing a link shared its bandwidth equally. Like previous simulation studies [BHP06, WC06] we assumed that bandwidth bottlenecks only occurred at the edge and did not model shared bottleneck links in the interior of the swarm.

We assumed that each downloading session consisted of a single source-server distributing the content to all peers. Unless otherwise specified we use the default settings summarized in Table 1. Unlike previous studies, we do not impose limitations on upload and download rates. Since we are assessing the fairness in BT we do not impose these limitations and assume that the fairness achieved gives sufficient incentive for all peers to fully dedicate themselves.

In [BHP06] all peers stay in the swarm until the last peer finished downloading the file. As BT offers no incentive for a peer to become a seed after downloading the whole file we observe the performance of the swarm over 25 minutes while the peers are leeches.

8.2 Responding to malicious behavior and seed attacks

In this set of experiments we considered a homogeneous setting where all peers within a network have the same link bandwidth of 1 Mbps. Randomly selected rogue peers in the simulation upload bogus chunks to other peers. These peers also act selfishly like rogue peers by downloading the file from seeds and other peers that unaware of their intentions.

Figure 1 relates to how efficiently our TMS is able to reduce the attacks by identifying the rogue peers. We varied the percentage of rogue peers in our simulations to see how they would affect the average percentage of file downloaded. The vertical lines extending above and below the columns represent the maximum and minimum values respectively.

As soon as a bogus chunk is downloaded for the local TMS simulations the rogue peer is choked. Although uploading bogus chunks, the rogue peers manage to stay in the swarm and it could be unchoked during the optimistic unchoking interval by other peers. As a result, unaware of its intention another honest peer may upload chunks to the rogue peer.

This situation worsens when a rogue peer is connected to the seed. In absence of global trust score the seed is unable to distinguish between an honest peer and a rogue peer. In such a case the rogue peer does not contribute anything to the swarm but is able to download the file for free by using the upload bandwidth of the seed that could have been used by other honest peers. Thus as seed cannot distinguish between rogue peers and honest peers it permits free riding and this slows down the data distribution process of the swarm.

In contrast, when TMS is used each rogue peer is able to upload first few bogus chunks before its global trust score is lowered thereby warning other peers that it is a rogue peer. Because of their negative trust scores these peers are not unchoked by other leeches and seeds. This minimizes the downloading by the rogue peers in the swarm. Furthermore as others peers including seeds prefer unchoking peers with larger trust scores this prevents the rogue peers from exploiting the seeds or the opportunistic unchoking

Table 2. Average file download (percentage)

	Average	Slow peers	Fast peers
BT	61.29	52.50	70.09
BT-local	60.11	47.59	72.62
BT-TMS	63.98	53.30	74.67

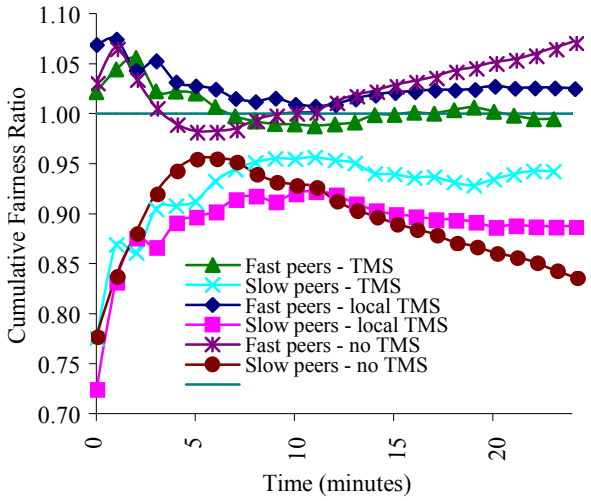


Figure 4. Effect of TMS on cumulative fairness ratio

mechanism. This also results in improvement of the performance of the honest peers (as shown in Figure 2).

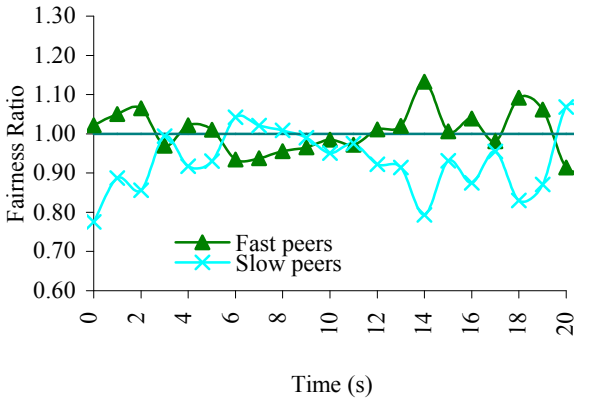
Figure 2 demonstrates the performance of the honest peers in the swarms when TMS is used as compared to when TMS is not used. We observe the average percentage of file downloaded when there are only an equivalent number of honest peers in the swarm and use it as a baseline. These are the consequences that we would guess given the behavior of the rogue peers in the simulation.

Finally in Figure 3 we observe computing trust score based on several chunk downloads i.e. a longer penalty interval may allow peers to make better decisions about whether or not to unchoke a peer. But the longer the penalty interval over which the trust score is computed the greater the vulnerability of the network to sudden subversion of previously honest peers or sudden change of tactics by rogue peers. It would take more time to rehabilitate a previously infected peer.

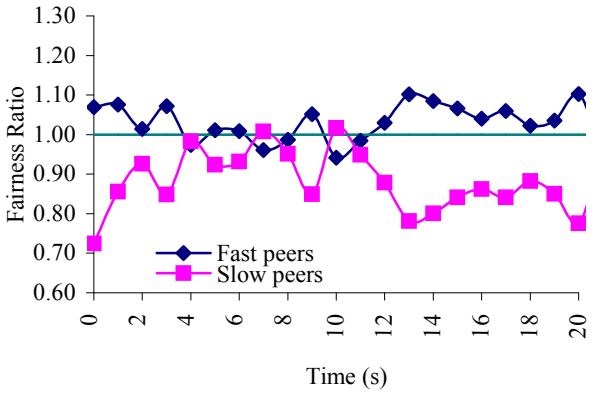
8.3 Ensuring fairness

In this set of experiments we study the behavior of BT swarms in a heterogeneous setting consisting of equal number of peers with 1 Mbps (*fast*) link bandwidth and 0.5 Mbps (*slow*) link bandwidth.

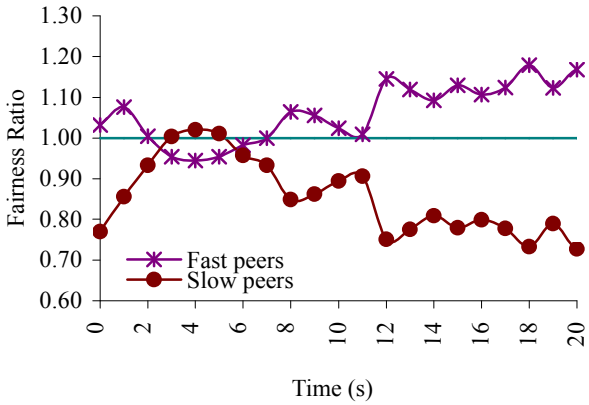
We first focus on the correlation between peer link bandwidth and the average percentage of file downloaded. Table 2 shows that peers with higher link bandwidth clearly download a higher percentage of the file. Using only the local TMS results in decrease in the upload capacity consumption because peers can potentially cease to upload



(a) BT – TMS



(b) BT – local TMS



(c) BT

Figure 4. Comparing fairness ratio versus time for fast and slow peers

while waiting to obtain reciprocal chunks. This is avoided by using global knowledge in TMS. Peers with TMS consider set of peers currently uploading to it. In addition it

prefers peers that are uploading to others in the swarm, as their upload bandwidth is unlikely to decrease. Consequently it helps peers find neighbors with similar bandwidth to it.

Next we measure *fairness ratio* in the terms of the number of chunks uploaded to that downloaded by each peer. Figure 4 presents the cumulative fairness ratio experienced by group of peers averaged up until that time. The peers are grouped together based on their link bandwidth. When only BT is used fast peers end up serving much more than they receive. This phenomenon would indeed worsen in presence of selfish peers as lack of reward and punishment can stimulate free riding that might make the swarm inefficient. By using TMS the fairness ratio is maintained close to 1 for both fast and slow peers. This indicates the presence of TMS significantly reduces the advantage that the peers with slow peers enjoy with BT. Furthermore using the global trust score matches peers with similar bandwidth and thereby results in much better fairness ratio compared to using local TMS only.

Figure 5 shows the fairness ratio averaged over group of peers in the swarm during the past 1 minute. The fairness ratio for BT (Figure 5c) tends to improve for the first 5 minutes after which the system enters a stable state where fast peers contribute more than the slow peers. The fairness ratio gets worse because the fast peers do not download as fast as they upload while slow peers benefit significantly when optimistically unchoked by the fast peers. When using TMS (Figure 5a) the fairness ratio tends to improve for the first 3 minutes after which it fluctuates about the best steady state fairness. Note that the number of chunks and not the rate is used to measure the fairness ratio. This causes the fluctuation in the curves in Figure 5 (a, b and c), which vary by significant amount over time.

9. CONCLUSION

BT and other P2P architectures rely on open and unsupervised chunk exchange among peers to overcome server bandwidth limitations and achieve very high data propagation rates. As a result, these architectures are vulnerable to more types of attacks than conventional client server solutions. We have presented a partially decentralized TMS tailored to the needs of the BitTorrent protocol.

Our trust management system requires each node to keep track of the quality of the data it receives from each of its neighbors and the overall fairness of the exchange process, thus allowing each peer to detect both rogue peers and selfish ones. In addition, our system uses the tracker of each BT swarm to compute and distribute global trust scores complementing the local trust scores maintained by each peer. Our simulation study indicates that our system significantly minimizes the downloading by a rogue peer in the swarm. We also found out that it resulted in a much fairer allocation of chunks among peers with different link bandwidths. Another major conclusion of our study is that the adoption of both global as well as individual trust knowledge is required to build resilient BT swarms.

Our work constitutes an important contribution to the BT protocol, as it will improve its ability to rapidly disseminate critical information in the presence of an active opposition. While our approach was tailored to the specifications of the BT protocol, it could also be applied to other partially decentralized P2P networks.

Future work includes investigating other possible subversion attacks, such as collusion attacks, and analyzing the robustness of our TMS.

REFERENCES

[AD01] K. Aberer and Z. Despotovic, "Managing trust in a peer-to-peer information system," *Proc. 10th Int. Conf. on Information and Knowledge Management*, New York, NY, 2001.

[BHP06] A. Bharambe, C. Herley and V. Padmanabhan, "Analyzing and improving a BitTorrent network's performance mechanisms," *Proc. IEEE 25th INFOCOM Conf.*, Barcelona, Spain, 2006.

[Bit07b] Multitracker metadata entry specification, *http://bittornado.com/docs/multitracker-spec.txt*

[BK05] R. Bazzi and G. Konjevod, "On the establishment of distinct identities in overlay networks," *Proc. ACM Symp. on Principles of Distributed Computing*, Las Vegas, NV, 2005.

[Bow07] P. Bowes, Warner to start movie downloads, *http://news.bbc.co.uk/1/hi/business/4753435.stm*.

[Coh03] B. Cohen, "Incentives build robustness in BitTorrent," *Proc. Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, 2003.

[Dou02] J. Douceur, "The Sybil attack," *Proc. Int. Workshop on Peer-To-Peer Systems*, Cambridge, MA, 2002.

[DV+02] E. Daminani, S. Vimercati, S. Paraboschi, P. Samarati and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," *Proc. ACM Conf. on Computer and Communication Security*, Washington DC, 2002.

[Fas07] FastTrack protocol, Available: *http://en.wikipedia.org/wiki/FastTrack.*

[FL+04] M. Feldman, K. Lai, J. Chuang, and I Stoica, "Robust incentive techniques for peer-to-peer networks," *Proc. 3rd ACM Conf. on Electronic Commerce*, New York, NY, 2004.

[GJA03] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," *Proc. 13th ACM Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, Monterey, CA, 2003.

[Gnu07] Gnutella protocol, Available: *http://www.gnutella.com.*

[JA05] S. Jun and M. Ahamad, "Incentives in BitTorrent

induce free riding," *Proc. Workshop on Economics of Peer-to-Peer Systems,* Philadelphia, PA, 2005.

[Kaz07] *http://www.kazaa.com.*

[KBC05] T. Kohno, A. Broido, and K. C. Claffy, "*Remote physical device fingerprinting*," IEEE Trans. on Dependable and Secure Computing, 2(2), 2005

[KSG03] S. Kamvar, M. Schlosser and H. Garcia-Molina, "The Eigen algorithm for reputation management in P2P networks," *Proc. 12th Intl. WWW Conf.*, Budapest, Hungary, 2003.

[LM+06] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, Free riding in BitTorrent is cheap. *Proc. Workshop on Hot Topics in Networks*, 2006.

[LN+06] N. Liokgas, R. Nelson, E. Kohler and L. Zhang, "Exploiting BitTorrent for fun (but not profit)," Proc. *International Workshop on Peer-to-Peer Systems*, Santa Barbara, CA, 2006.

[MG04] S. Marti and H. Garcia-Molina, "Limited reputation sharing in P2P systems," *Proc. ACM Conf. on Electronic commerce*, New York, NY, 2004.

[Par04] A. Parker. *The true picture of peer-to-peer file sharing, http://www.cachelogic.com/*

[RZ+00] P. Resnick, R. Zeckhauser, E. Friedman and K. Kuwabara, "*Trust management systems*," Communications of the ACM, 43(12), 2000.

[SF+04] N. Stakhanova, S. Ferrero, J. Wong and Y. Cai, "A reputation-based trust management in peer-to-peer network systems," *Proc. Intl. Workshop on Security in Parallel and Distributed Systems*, San Francisco, CA, 2004.

[SG+02] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble and H. Levy, "An analysis of Internet content delivery systems," *Proc. USENIX Symp. on Operating systems Design and Implementation*, Boston, MA, 2002.

[SP+07] M. Sirivianos, J. Park, R. Chen and X. Yang, "Free-riding in BitTorrent networks with the large view exploit," *Proc. Intl. Workshop on Peer-to-Peer Systems*, Bellevue, WA, 2007.

[TC05] R. Thommes and M.J. Coates, "BitTorrent fairness: analysis and improvements," *Proc. Workshop Internet, Telecom. and Signal Proc.*, Noosa, Australia, 2005.

[Tra07] Trackerbt - a distributed BitTorrent tracker, *http://www.alhem.net/project/trackerbt/index.html*

[Wag07] J. Wagner, Lindows Grabs BitTorrent By The Bit, *http://www.internetnews.com/dev-news/article.php/3321911*.

[WC06] G. Wu and T. Chiueh, "How efficient is BitTorrent?" *Proc. 2003 SPIE Multimedia Computing and Networking Conf.*, San Jose, CA, 2006.

[WP02] A. Wagner and B. Plattner, "Peer-to-peer systems as attack platform for distributed denial-of-service," *Proc. ACM SACT Workshop*, Washington, DC, 2002.

[WS05] K. Walsh and E Sirer, "Fighting peer-to-peer spam and decoys with object reputation," *Proc. 3rd Workshop on the Economics of Peer-to-Peer Systems*, Philadelphia, PA, 2005.

[XL04] L. Xiong and L. Liu, "*Peertrust: supporting reputation-based trust in peer-to-peer communities*," IEEE Transactions on Knowledge and Data Engineering, Special Issue on Peer-to-Peer Based Data Management, 16(7), 2004.

[YA05] W. Yang and N. Abu-Ghazaleh, "GPS: a general Peer-to-Peer simulator and its use for modeling BT," *Proc. 13th Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Atlanta, GA, 2005.

[ZH06] R. Zhou and K. Hwang, "*PowerTrust: a robust and scalable reputation system for trusted peer-to-peer computing*," IEEE Transactions on Parallel and Distributed Systems, 18(5), 2007.

[ZL04] C. Ziegler and G. Lausen, "Spreading activation models for trust propagation," *Proc. 2004 IEEE International Conf. on e-Technology, e-Commerce, and e-Service*, Taipei, Taiwan, 2004.

Purvi Shah received the BE (2001) and MS (2003) degrees in computer science and engineering from University of Pune in India and University of Alabama at Birmingham respectively. She is currently working on her doctoral dissertation on P2P multimedia solutions at the University of Houston. During the summers of 2006 and 2007, she was a research intern with the Hewlett-Packard Labs, Palo Alto, CA and in summer 2005 she was a research intern with the Xerox Innovation Group, Webster, NY. Her research interests include the research and development of network protocols and distributed systems.

Jehan-François Pâris is a Professor of computer science at the University of Houston. He was formerly on the faculty at Purdue University and the University of California, San Diego. Dr. Pâris received his Ingénieur Civil degree from the Université Libre de Bruxelles, Belgium, his Diplôme d'Etudes Approfondies from the Université de Paris VI, France, his Licence et Maîtrise en Informatique from the Facultés Universitaires de Namur, Belgium and his PhD in EECS from the University of California, Berkeley. His research interests include memory hierarchies, scalable reliable multicast protocols, distribution protocols for video-on-demand, and distributed systems security. He is a member of ACM and a senior member of the IEEE Computer Society.