Defending BitTorrent against Strategic Attacks

Darshan Purandare, Ratan Guha and Abeer Hamdy

School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, USA
E-mail: {pdarshan, guha, abeer}@cs.ucf.edu

Abstract: BitTorrent has shown to be efficient for bulk file transfer, however, it is susceptible to free riding by strategic clients like BitTyrant. Strategic peers configure the client software such that for very less or no contribution, they can obtain good download speeds. Such strategic nodes exploit the altruism in the swarm and consume resources at the expense of other honest nodes and create an unfair swarm. More unfairness is generated in the swarm with the presence of heterogeneous bandwidth nodes. Many high capacity peers contribute much more than needed while low capacity peers contribute very little or nothing. In this paper, we propose and investigate new anti-strategic policies that could be used in BitTorrent to minimize the free-riding by strategic clients. In our proposed anti-strategic model, nodes obtain a *token* from Tracker upon joining the swarm which they use while interacting with peers. The token contains information such as *published upload speed*, arrival time and node ID, and this token is signed by tracker such that other nodes can verify the information but nobody can forge it. Other anti-strategic policies include, using a smart tracker that denies the request of strategic clients for peer list multiple times, and black listing the non-behaving nodes that do not follow the protocol policies. These policies help to stop the strategic behavior of peers to a large extent and improves overall system performance. Moreover, in this paper, we also quantify and validate the benefits of using bandwidth peer matching policy. Peers are given a peer list based on their bandwidth range upon arrival. This fosters better uplink utilization, reduces the time for nodes to find the optimal peers for exchanging data and has positive effects on many important metrics like download time, fairness index etc. Our simulations results show that with the above proposed changes, uplink utilization and mean download time improves considerably. It leaves strategic clients with little or no incentive to behave greedily. This reduces free riding and creates fairer swarm with very little computational overhead. Finally, we show that our model is self healing model where user behavior changes from selfish to altruistic in the presence of the aforementioned policies.

Keywords: Peer-to-Peer, BitTorrent, Content Distribution, Strategic clients

1 INTRODUCTION

BitTorrent (BT) (6) has emerged as one of the most popular peer-to-peer (P2P) models in recent years for bulk file sharing. It shows improved performance in terms of uplink utilization and mean download time as compared to other P2P systems (3). However, BT is prone to strategic attacks as shown by BitTyrant (17) and others (8; 12; 13). It also suffers from free riding problem (2) and creates unfair swarm. Many high capacity peers[a] upload much more than it is required while many get a free ride. The share ratio (ratio of uploaded volume to downloaded volume content) has shown to vary from 0 to almost 6 in many studies (3).

The strategic clients exploit the excess bandwidth in the swarm provided by some altruistic peers present in the swarm. They game their BT client in such a way that with very little or no contribution, they can obtain a good download speed. This exploit comes at the expense of other non strategic users. They procure a peer list in excess of 200-300 by constantly querying for more peers from the tracker. Their main strategy is to exploit the optimistic unchoke by seeds, and if there are many seeds, they could benefit without contributing much. While interacting with other nodes in the swarm, these clients gradually decrease their uplink speed while they get service from other peers. If the other peer drops the service because of less uplink speed, such strategic clients increase the uplink speed so as to reach the minimal uplink speed needed to induce cooperation from the other peer. Such clients use many more TCP connections than mentioned in the reference BT client implementation to exploit the maximum download speed for a given uplink speed.

In this paper, we propose a set of new features that, when incorporated, could make BT resistant to such strategic attacks. We determine the impact of these policies on other important factors in the system like mean download time, uplink utilization and fairness towards an end user. There is a trade off involved in accomplishing these goals simultaneously, i.e., minimal download time and fairness do not go hand in hand and pursuit of one affects the other. The BT protocol can be customized in many different ways, where achieving optimal mean download time is one end of the spectrum and achieving fairness the other (4). We investigate as to what point in the whole spectrum of these parameters, could yield a near optimal results with no strategic attacks and high fairness. We also investigate the effect of altruism, i.e., self volunteers who offer their upload speed in return for nothing, on the swarm performance.

The main goals of BitTorrent like P2P system are the following. Each one of them is very important independently and not completely mutually exclusive with others. In our present work, we propose new policies that will make them robust and capable of overcoming the current problems of strategic attacks.

- Survivability: Every block of the file must exists in the swarm at all times so as to ensure that all nodes can finish the download at some point of the time, better sooner. BT employs *local-rarest-first* policy for replication of pieces and has shown to be very efficient (11).
- Download Time: Every node individually attempts to finish its download as soon as possible. Selfish clients use greedy policies for the same. Our aim is to keep the mean of download times of all the nodes to as less as possible.

[a] We have used the terms nodes and peers interchangeably.

- Uplink Utilization: High uplink throughput is desirable for scalable system and partially reflects the mean download time. Peers at every point of time attempt to find a partner which has high uploading capacity. Seeders upload the content to the peers with high download speed to effectively improve the uplink throughput, which improves download time. In later and new version of BT, the seeds uniformly distribute the content to the nodes rather than few high capacity nodes.
- Fairness: The swarm should be fair i.e., no node should be forced to upload much more than what it has downloaded. No one should be able to get a free ride. Voluntary altruism is welcome for the swarm.
- Robust: The swarm should be robust to strategic clients and must not let these clients to download selfishly at the expense of other non strategic nodes.

In this paper, we propose new features for BT to overcome strategic attacks and to improve the overall system performance. In particular our main contributions and findings are:

1. We use anti-strategic policies to guard BT against selfish clients (17; 13). To this end, peers exchange tokens given to them by tracker for keeping tab of uplink speed of the other peer. Intelligent tracker prohibits strategic clients from procuring peer list multiple times. Nodes that upload garbage content are quickly identified and blacklisted by neighboring nodes. These policies are resistant to strategic attacks and does not let the clients degrade performance of other peers.
2. We quantify and experimentally validate the concept of bandwidth peer clustering (19; 3; 11) and show that it shows significant improvement in uplink utilization and mean download time.
3. We show that altruism is indeed very important for improving the overall system performance. Altruistic swarms finish the download much faster and use the resources near-optimally when compared to swarms without altruism.

Section 2 gives an overview of BT like P2P model. We present the details of our model in section 3. Section 4 describes the metrics we use for evaluating our model and the simulation setup. We briefly describe the metrics which we use in our simulations and comparisons. And finally results are in section 5. In section 6, we present the related work done for the enhancement of the BT protocol.

2 BitTorrent Details

BitTorrent (BT) (6) is a P2P application used for bulk file download. BT leverages the uplink speed of peers to efficiently replicate content among a large set of peers. In BT, files are broken into small pieces typically 256 Kb each. As the fragments are distributed to the peers in a random order, they can be reassembled on a requesting machine. To share a file using BT, the publisher (seed) creates a .torrent file, a small metafile that contains the information like filename, size, hash of each block in the file, the address of a tracker server and miscellaneous data like

client instructions. The .torrent file is distributed to the users via some medium like email or website. The original user who is willing to offer the upload starts as a seed while other users start as leeches. Once a new user procures the .torrent file and joins the system, it contacts the tracker to obtain a list of 40 peers (6) including seeders and leeches that are in the swarm. A new node upon receipt of peer list from tracker contacts these nodes to obtain the file blocks. The nodes in the peerlist which are already in the system send their buffermaps to this new node. Buffermap is a list which contains the list of pieces they currently have. New node then requests the pieces from these nodes. If the peer list goes below 20 due to departure of some nodes, the new node makes another request to tracker for additional peers. Each peer periodically, typically 30 minutes, reports to the tracker the number of pieces it has procured. The helps the tracker to make informed decisions while assigning a peer list to a new user. In case if a node is departing from the swarm, it intimates the tracker about the same.

Every node attempts to download as many pieces and as fast as it can. For each available source, the node considers the blocks of file available and then requests the rarest block (least replicated) among the local peers. This is called as Local Rarest First(LRF) policy. The rarest block is chosen and downloaded to maximize the content diversity in the system. This makes it more likely that peers will have blocks to exchange. As soon as the client finishes importing a block, it hashes the block to ensure that the hash matches with the hash value in the torrent file. It then sends a *have* message to its neighbors about this new piece, so that other nodes can now request for this newly obtained piece.

BT uses *tit-for-tat* policy to ensure that nodes in the system not only download the content but also contribute by uploading their pieces back. Every node periodically unchokes i.e. allows a small number, say 5 (6), connections for outgoing link to avoid having lots of competing TCP connections (3; 16). Every node keeps track of the pieces they obtain and the speed at which they obtain those pieces from their neighbors. A node gives its uplink speed to only those nodes that give it the best possible connections in terms of uplink speed. This property of BT is strategic resistant, i.e., it mandates the peers to upload pieces in the swarm to get downloads from neighbors. The decision to select which nodes to give uplink connection is taken every 10 seconds. Every 30 seconds, nodes in the system periodically select a random peer in its neighborhood to give its uplink connection. This process is called optimistic unchoke. It helps newly arrived users in the swarm to get started and nodes can periodically evaluate and find if there is any peer that can offer it a better uplink speed than its present set of active connections.

This process of uploading and downloading continues till nodes procure a complete copy of the file. Once a node finished it can either stay in the swarm and offer its uplink to helps others or it can leave the swarm. The average download time of the users in the swarm is proportional to the number of seeds present and also on the contribution of other nodes.

2.1 Strategic Attacks on BitTorrent

P2P systems are inherently based on user altruism and participation (3) but this concept is exploited by free riders who do not want to contribute their uplink speed and are only interested in downloading. Free riders in BT resort to

acts like tuning their client for minimum upload or even strategizing the client for maximal download for minimum contribution. Liogkas et al. (12) proposed 3 main techniques to exploit BT: 1) Download from seeds. 2) Download from peers with high uplink capacity and 3) Advertising false pieces. Locher et al. (13) proposed to procure huge peer list so that a client can afford to only interact with the seeds. BitTyrant (17) is a client implemented using Azureus (Azu), and has been very successful in exploiting the vulnerabilities of BT. BitTyrant exploits the altruism present in the swarm by procuring a large peers list in the swarm. It then attempts to establish connections with seeds and procures free content. While interacting with other nodes, BitTyrant adopts a policy of uploading minimum content to get the maximum download. One of the very important exploits BitTyrant used is to avoid equal split policy while uploading, rather it uploads in different fractions to its peers to get the maximal download. While interacting with a peer, it gradually decreases its uplink speed as long as it gets reciprocation. If the other peer chokes BitTyrant then it increases the uplink speed and determine the level of participation (uplink speed) needed to keep the connection alive and get the downloads. BitTyrant does this over many connections and attempts to maximize the download for a given uplink speed limit. BitTyrant by virtue of its greedy policy does bring performance improvement for a client, but it can hurt the swarm performance if all the peers use the BitTyrant client (17). This would mean BitTyrant improves performance for an individual user and not for the whole swarm. In this paper, we investigate if such strategic behavior could be alleviated in the swarm while achieving near-optimal behavior in the overall swarm performance. To this end we propose new set of anti-strategic policies mentioned in the next section.

3 Our Proposed Policies

In this section, we propose a set of features for BT that could help to overcome the strategic attacks and improve the swarm performance.

3.1 Anti-Strategic Behavior

We propose to use anti-strategic behavior in every BT client in the wake of recent attacks which leads to poor swarm performance (17). In particular, we propose to use the following three strategies to overcome strategic attacks.

3.1.1 Token From Tracker

We introduce the notion of *Published Upload Speed* (PUS) for a node. This is to dissuade the node from cheating and to ensure that the node offers almost same upload speed as published throughout the time it is in the swarm. Every node upon arrival submits to the tracker the PUS it is going to offer to the peers. The tracker creates a token, called *TokenFromTracker* ($[Node\ ID,\ PUS,\ Time]K_R$), which consists the Node ID, PUS, and its arrival time, and encrypts it with its private key (K_R). While interacting with peers in the swarm, nodes exchange their $TokenFromTracker$. By decrypting the other peer's token ($\{[Node\ ID,\ PUS,\ Time]K_R\}K_U$)

using the public key of tracker (K_U), they get an estimate of the uplink speed the other node is offering.

Assuming a node splits its uplink bandwidth equally among its local peers (say 5), a node can gauge which peers give better uplink speed and it can unchoke the connections to those peers. While a session is on, nodes can resort to cheating by initially offering uplink speed equal to PUS and later on gradually decreasing the uplink speed (similar to BitTyrant). A node can be immediately caught if it keeps upload bandwidth low for a certain period of time interval. The model associates some tolerance T with the uplink speed i.e. if a node publishes an upload speed of U, then its uplink speed in the range $(U - T)$ to U is acceptable. If it falls below $(U - T)$ the other node would wait for a small time interval t and eventually disconnect. The tolerance T and t takes care of the network anomalies that might arise and a non strategic user should not be punished for it. Peer notifies the tracker about the cheating node. Tracker warns such nodes after h complaints (say $h = 3$) and upon receiving k such complaints (say $k = 5$), tracker bars the node from getting any more peer list and blacklists such node, and intimates the nodes in its peer list about the existence of such cheating node. This is a consequence of the punishing policy of our model. Nevertheless, the warned users who cooperate with the protocol, i.e. after warning stop the strategic behavior are allowed to stay in the swarm. This is in accordance with our self healing policy.

An end user (node) can configure the BT client based on its preferences. A node can have a good uplink speed, but it would not want to dedicate all its uplink speed to BT application. The node can resort to cheating by initially offering high PUS and later configuring its BT client to downgrade the uplink speed. Open source programs for BT give an end user a chance to modify the protocol in the code. New programs like Azureus (Azu) can be customized as per user needs and allow user to choose upload speed, number of connections etc. This facilitates cheating by an end user. If a node wants to downgrade its PUS for some reasons, it is expected that the tracker is informed and the node obtains a new $TokenFromTracker$ with the new downgraded PUS. This helps the tracker differentiate the cheating nodes from the non cheating ones. Nodes intimate the tracker while leaving the swarm.

3.1.2 Smart Tracker

Most strategic attacks stem from the fact that such clients request for more peers from the tracker. After every small time interval, they request for additional peers till they have peers in excess of 200-300. They then launch strategic attacks by interacting with more peers and also hope to get optimistic unchoke from high capacity peers. We propose to use a *smart tracker*, wherein for every request for additional peer list by a client, tracker checks if it has 40 live peers in the swarm. If no, it provides it additional peer list so that the client has at least 40 peers in the list, else it rejects its request of additional peer list. This can substantially lower down such strategic moves by clients who are trying to maximize the profit. In the results section, we quantify the effectiveness of smart tracker in alleviating the behavior of such clients.

3.1.3 Blacklisting Nodes

To counter the false publishing attack of nodes wherein the client falsely sends 'have' messages of rare blocks in the swarm and in turn uploads garbage content in order to obtain some useful content, we propose to use a policy wherein a node blacklists such a node upon finding that it uploaded garbage content. Future interactions with such nodes are avoided and tracker is made aware of such garbage content. Tracker warns such nodes after h complaints (say $h = 3$) and upon receiving k such complaints (say $k = 5$), bars the node from getting any more peer list and blacklists such node. The selfish nodes in presence of such policy cannot publish false message for long as they will be blacklisted by most nodes in its neighborhood and it will severely hurt its chances of obtaining new and useful content. In the results section, we show that by adopting this policy, significant swarm bandwidth, that goes in downloading such garbage content, can be saved. It also serves as a warning to such nodes that publishes false content and upload bad content not to resort to such techniques. Results show that user behavior indeed changes from selfish to normal upon receiving warning from tracker.

3.2 Classification of Nodes in bandwidth stratum

We propose a scheme to classify and group nodes which are in similar bandwidth ranges (19). Small et.al. (22) has showed that nodes with similar bandwidth capacities when interact improve the uplink throughput and hence increase the scalability of the swarm. Nodes upon arrival submits the PUS to the tracker. Based on its PUS it is placed in certain stratum in which its bandwidth lies. Upon arrival a node is handed a peer list which contains more nodes from its bandwidth range. It is also given peer list from other bandwidth stratum so that a node can help or get helped from other high or low capacity strata and the flow of content is maintained all throughout in the swarm.

In our simulation, we consider 70% peers from the same stratum and 30% from other strata. It can be denoted as $(70, 30)$. We also consider other possible distributions like $(60, 40)$ and $(50, 50)$ etc. We experimentally obtained these distributions which is mentioned in section 5.3. Later, we show that this scheme of assigning peers is better than random selection of peers (used in conventional BT model). It ensures near optimal uplink utilization as maximum interaction takes place among the nodes within the same strata. Moreover, this scheme tends to be fair as the nodes form a symbiotic association and are equally benefited in terms of the volume of content served. This forms a natural incentive for the nodes to be in the best possible strata and encourages to publish and contribute maximum of their uplink bandwidth capacity.

4 Evaluation of Our Proposed Model

We present the details of the simulation setup for our proposed anti-strategic and strata based model. We present an evaluative comparison against the BT protocol. The main metrics for comparison are average time to finish the download, uplink bandwidth utilization, and fairness index in terms of the share ratio. We now detail

the metrics used for our comparison.

1. Average download time: It is the mean of download times of all the nodes in the system. Mathematically,

$$Average\ Download\ Time = \frac{\sum_{i=0}^{N} D_i}{N}$$

where D_i is the download time of node i and N is the total number of nodes in the swarm. The download time of high capacity nodes has shown to be less compared to its weaker counterparts (3). By evaluating the mean download time for the whole swarm, we can get a fair bit of idea about the performance of the BT protocol in the swarm.

2. Uplink utilization: Uplink bandwidth is the most sparse resource in the system. In most realistic scenarios $Uplink\ Bandwidth \leq Downlink\ Bandwidth$ holds good, so we limit our discussion to only uplink bandwidth. A good uplink throughput i.e. (ratio of uplink used to uplink available) would mean a lot of resource (uplink bandwidth) are pooled in the swarm that can serve peers, which in turn helps to lower the download time. Moreover, some ISPs charge their end users for the bandwidth used per unit time. Such users would want to maximize the uplink throughput for saving the ISP fee. Small et al. (22) proved that maximization of uplink speed leads to scalable systems. Mathematically,

$$Uplinl\ Utilization = \frac{\sum_{i=0}^{N} UT_i}{N}$$

where UT_i is the ratio of the uplink bandwidth used to the uplink bandwidth available for node i.

3. Fairness: We define fairness in terms of share ratio of content served by nodes. Share ratio of end users over the period of complete download depicts the contribution of the nodes quite fairly. In an ideal system, nodes have share ratios of 1.0, where an end user downloads the content and passes on equally to other end users. But given the dynamics of the internet, churn, and peering schemes, it is very difficult to achieve the same in BT. So, more the number of nodes close to share ratio of 1.0, the fairer is the system, i.e., the lesser the variance of share ratios from the mean, fairer is the system. We use Jain's fairness index (10) to evaluate the swarm fairness.

$$f(x_1, x_2, x_3, \ldots, x_n) = \frac{(\sum_{i=0}^{n} x_i)^2}{n \sum_{i=0}^{n} {x_i}^2}$$

where $x_1,\ x_2, ...,\ x_n$ are the share ratios of the nodes. The value of fairness index varies from 0 (worst) to 1 (best).

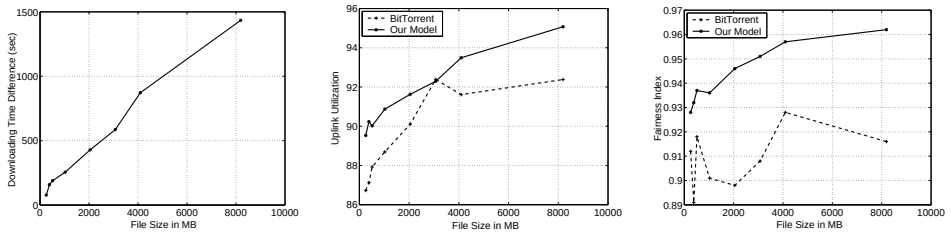
4. Altruism: We define altruism as the excess uplink bandwidth provided to the swarm by a node, i.e., a node when willingly uploads more content than what it downloaded is altruism for the swarm. Most altruistic behavior is displayed by the original seed. Altruism can be either voluntary or sometimes circumstantial/forced. Voluntary altruism is welcome as it helps the system with more resources and the load is divided. In forced altruism, a node has to upload more content to even download a single copy of the file (torrent data). Forced altruism is not fair as a node is compelled to upload more than what it had downloaded. We define Altruism as the excess content uploaded to the swarm after reaching a share ratio of 1.0. For e.g., if a node has uploaded 6 copies and downloaded 1, its Altruism factor will be 5.0 since it uploaded 5 excess copies of the torrent data in the swarm. A node with share ratio of 1.0 will have altruism factor as 0. Mathematically,

$$\begin{aligned} Altruism\ Factor &= \frac{Uplaods - Downloads}{Downloads} \\ &= Share\ Ratio - 1 \end{aligned}$$

4.1 Simulation Setup

Most real world physical links have ($downlinkspeed > uplinkspeed$), so the bottleneck in most cases is the uplink speed. In such cases downlink speed cannot capture the correct notion of bandwidth utilization. To justify our fairness claims, we have taken into account share ratio of the nodes in the system. We compute the fairness index of share ratios and compare with BT to identify as to how our model works in case of free riders. Can we lower the disparity of share ratios of the nodes so that free riders have no incentive in their behavior? We have performed experiments to evaluate our self healing and self punishing model. We used a simulation based approach, primarily because it is extremely difficult to gauge the behavior of large swarms without the participation of thousands of node. Moreover, simulated settings give flexibility to play with different parameters without affecting the overall behavior.

We used the BRITE universal topology generator (15) in the Top-Down Hierarchical mode to model the physical network topology of Autonomous Systems (AS) and the routers. All AS are assumed to be in the Transit-Stub manner. Overlay is assumed to be undirected. Unlike other simulators (3; 14), we assume that the bottleneck in the network can appear in the access links of source and destination (i.e. first-mile and last-mile hops) as well as the non access links that are in the interior of the network, in particular within or between carrier ISP networks. The nodes in the swarm are assumed to be of heterogeneous bandwidth classes namely:(512Kb, 128Kb), (768Kb, 256Kb), (1024Kb, 512Kb), (1536Kb, 768Kb), (2048Kb, 1024Kb) where first and second member of the tuple are the maximum downlink and uplink speed of a node respectively. The distribution of these bandwidth classes is uniform in the swarm. To simulate the congestion in the Internet, we induce 5% congestion in the non access links within the interior of the network. In such congestion scenarios, the available bandwidth to nodes is the minimum of the bottleneck at source or destination and the bottleneck in the non access links. The delay on



(a) Download Time Difference vs File Size in MB (b) Uplink Utilization vs File Size in MB (c) Fairness Index vs File Size in MB

Figure 1 *Effect of Anti-Strategic Behavior in Our Model*

inter-transit domains and intra-transit domains are assumed to be 100 ms and 50 ms respectively, while delay on stub-transit is assumed to be 30 ms and intra-stub transit links are randomly chosen between 5ms and 25ms. We simulate the TCP level dynamics like timeouts, slow start, fast recovery and fast retransmission by introducing a delay of 10 RTTs (5). We model a flash crowd scenario for the arrival of users in the swarm, i.e. all users are present in the swarm when the file sharing begins, as this is the most relevant and challenging scenario.

In our simulation setup we have varied the following parameters: Number of users (N) from 128 to 8192, File size (S) from 256 MB to 8192 MB. Each file block is considered to be 256 KB. Initial seed is considered to be a powerful node capable of very good upload speed say (6 Mbps). A default implicit assumption is that as nodes finish their downloads they leave the swarm. For some other experiments, we assume that nodes stay in the swarm. We mention the settings at the appropriate places in the text. Further, to evaluate our self healing and punishing model, we injected around 10% cheating nodes. These nodes mimic real world cheating nodes that do not adhere to the protocol. Implementation program detects such cheating nodes for their selfish behavior. We analyze the node behavior during the course of the simulation and quantify the number of nodes that turn from selfish to altruistic.

5 Results and Discussion

We use the above simulator to test the efficacy of our proposed policies. We compare the behavior of conventional BT protocol with our proposed changes. Our proposed changes are to overcome the current day vulnerability in BT which the strategic clients exploit. We test each of our proposed techniques one by one, and then later combine all the proposed changes to see the overall behavior.

5.1 Effect of Anti-Strategic Behavior

Current day exploits by strategic clients are: 1) Requesting peer list from tracker after every few minutes. 2) While interacting with other nodes, decrease the contribution gradually to arrive at an equilibrium that will give client maximum download speed for a given upload contribution. 3) Sending false 'have' messages and uploading garbage content. In the first set of experiments, we use the anti-strategic policy to see how well a swarm can cope up with the strategic clients. The main compo-

nents of our anti-strategic policy are mainly using *TokenFromTracker* that guards honest nodes from such selfish clients that degrade the upload quality, stopping selfish clients from obtaining peer list every few minutes by using a smart tracker and finally blacklisting the nodes that uploads garbage content.

In this set of experiments, we investigate the difference in average downloading time between the conventional BT with our proposed policies. Figure 1(a) depicts the results where the torrent file size varies from 256 MB to 8192 MB for a swarm of 1024 nodes. 10% of the nodes behave strategically all the time during the simulation in both the cases. For smaller file sizes the difference is small but with increasing file sizes the difference in download time is very prominent. For 8192 MB file, the difference in average downloading time is as high as 1450 seconds (approximately 40 minutes). Since every node keeps a tab on each of its connection, and as selfish clients degrade the uplink bandwidth, the other node disconnects leaving the selfish client with no option but to increase its contribution. We found that by using *TokenFromTracker*, the BitTorrent *Tit-For-Tat* mechanism is used effectively and the equilibrium of uplinks bandwidths between the nodes is reached earlier as compared to standard BT implementation. Legout et al. (11) and Piatek et al. (17) have showed that BT takes unusually long time to reach the steady state or equilibrium, where peers have found their optimal partners with respect to the uplink bandwidths. This means that non strategic clients do not suffer on account of strategic clients while using *Token From Tracker* as all legit nodes can interact with nodes which are in similar bandwidth range. In the download of a large file like 8192 MB with a large swarm, the difference in download time is evident as the policy of our model are enforced for a longer time. Similarly, the smart tracker denies the recurring requests of strategic clients for more peer list. In this case such strategic client have to make connections within the given peer list and they cannot exploit the best connections through seeds and high capacity leeches. Finally, the strategic nodes that send false 'have' messages for rare blocks for obtaining some useful content are quickly identified by fellow peers by checking the hash of the blocks. Such nodes are warned and denied any help from tracker in obtaining new peer list in future. Over a longer period of time, such nodes are found to adhere to the protocol. All the above factors show a considerable performance improvement in the average download time.

Similarly, the difference in uplink utilization can be understood from figure 1(b). Average download time of the swarm is directly proportional to the uplink utilization. By strictly imposing the anti-strategic policy on the strategic clients, we can leverage the uplink bandwidth of such selfish nodes. We see a consistent superior performance of our model over BT by using the anti-strategic policy. Our model on an average has $3 - 4\%$ more utilization than the BT. This difference amounts to large chunk of bandwidth (approximately 21 MB/s) for a 1024 node swarm and an average uplink bandwidth of 537.6 KB/s (average of all the uplink bandwidths of all the strata in the swarm). The more the uplink bandwidth is pooled in the swarm, the better will be the uplink throughput and hence better will be the average download time.

Figure 1(c) shows the fairness index of both the models. The FI has been calculated on the share ratios of the nodes in the swarm. If all the nodes have share ratio 1.0, then the system would be ideal and FI would be 1.0. But as the share ratio deviates from 1.0 the FI goes down. It gives a decent measure of the performance

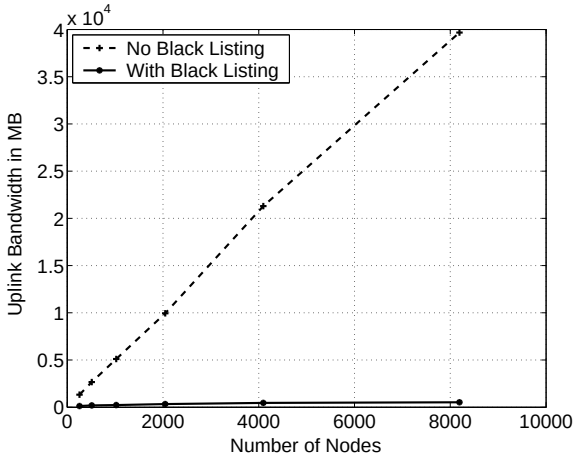


Figure 2 A comparison of bandwidth saved using blacklisting policy

of nodes in terms of contributing to the swarm. Strategic clients consume lot of bandwidth of the swarm with no or little contribution, compelling honest nodes to contribute more. This disparity of share ratio is quite evident in the figure 1(c). By using anti-strategic policy, we can eliminate the greedy behavior of selfish clients. The share ratios of the nodes are more even in case of our model. It is extremely difficult to achieve a perfect FI of 1.0, because of node heterogeneity, network topology, churn and altruism. We discuss the effect of altruism in section 5.5 as it has a major impact on the results.

5.2 Effect of Blacklisting

We quantify the amount of uplink bandwidth that can be saved by simply using blacklisting policy as mentioned in section 3.1.3. Figure 2 depicts an estimate of bandwidth that is saved in a 1024 node swarm with 10% strategic clients. We assume that such clients on an average upload 5% garbage data. For large torrent file size, the difference in bandwidth is huge, and it adversely affects uplink utilization and hence mean download time of the swarm. While in the case of anti-strategic policy, part of the bandwidth is lost until all the nodes that upload garbage are caught and blacklisted. Tracker informs peers of such cheating node not to interact with them. This indeed saves a huge chunk of the bandwidth.

In the next set of experiments, we started out an experiment with 1024 nodes out of which 900 are honest and 124 are strategic. During the simulation run, these nodes behave strategically. Upon being caught and warned by tracker, such nodes either adhere to the protocol or continue cheating. If they adhere to the basics of the protocol, they are included as honest nodes but upon constant cheating they can be blacklisted and thrown out by the tracker. Figure 3 depicts the results. Towards the end of the simulation run, 91 our of 124 strategic nodes turned honest and 31 were blacklisted. By giving strategic clients a chance to improve their behavior, we can leverage their contribution for the rest of the download. Some clients that consistently cheat are thrown because of their leech behavior, they do

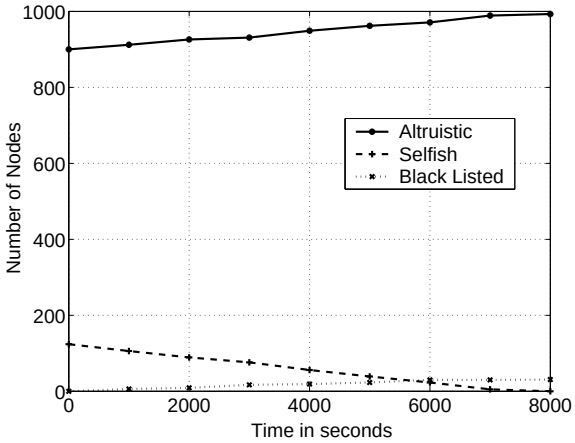
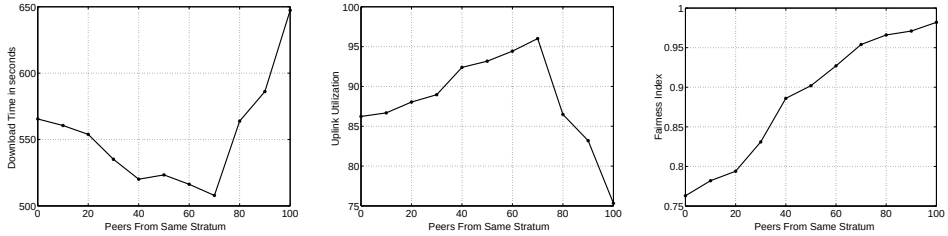


Figure 3 An estimate of nodes that turned from selfish to altruistic upon warned



(a) Download Time Difference vs No. of Nodes in same stratum

(b) Uplink Utilization vs No of. Nodes in same stratum

(c) Fairness Index vs No. of Nodes in same stratum

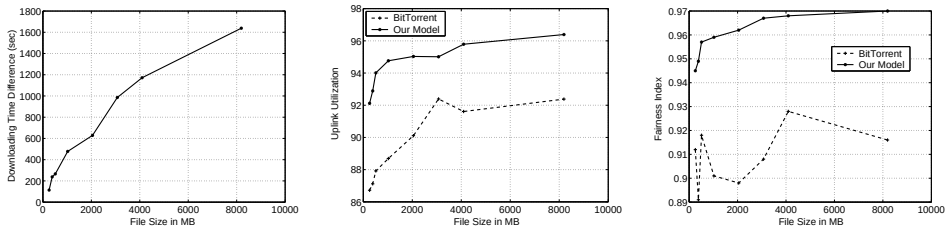
Figure 4 *Effect of Bandwidth Clustering in BitTorrent*

not contribute any resources to the swarm, which is against the ethics of the P2P file sharing etiquettes.

5.3 Effect of Bandwidth Clustering

Figure 4 depicts the results of bandwidth clustering in different settings. A tracker while assigning a peer list to the nodes, issues it a peer list which consists of some nodes from its own stratum and others from all other strata. In conventional BT, a tracker randomly issues typically 40-50 nodes without any consideration of uplink bandwidth of nodes. When two nodes of equal bandwidth interact, they leverage the full uplink bandwidth of each other. On the contrary when two heterogeneous nodes interact, one of the nodes could be a bottleneck in which the maximum uplink bandwidth is not used. We proposed to modify the BT scenario such that nodes often get to interact with peers of equal bandwidth capacity. This would improve the uplink throughput and hence download time (22). In this set of experiments, we determine the percent of nodes in the peer list from the same stratum.

In this set of experiment, we only focus on the issue of bandwidth clustering and



(a) Download Time Difference vs File Size in MB (b) Uplink Utilization vs File Size in MB (c) Fairness Index vs File Size in MB

Figure 5 *Overall effect of our Model*

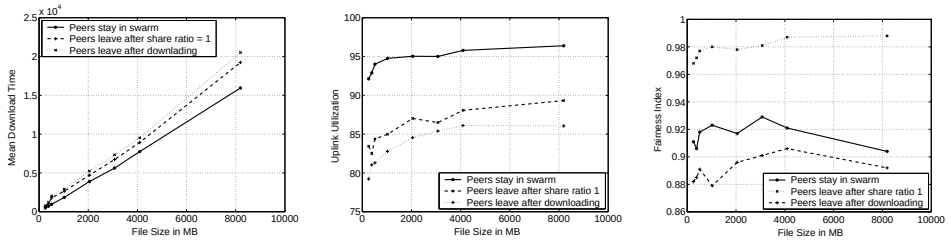
we assume no malice from strategic clients. A torrent file of 256 MB and a swarm of 1024 nodes is used for this setting. From figures 4(a), and 4(b), it is evident that uplink and hence download time are optimal around 60 to 70%. After 70% mark, the uplink and mean download time both show a sub optimal performance. The obvious reason is that since a node receives a peer list dominantly from its bandwidth stratum, it misses out on some important useful connections and pieces of blocks that might be present in the other stratum. In such case, many nodes are idle for a while as they don't have any interesting blocks to exchange. This lower the uplink throughput and hence mean download time. We found that $(70, 30)$ as an efficient peering policy for optimal swarm behavior, i.e. 70% nodes in the peer list from the same stratum and 30% from other strata.

From figure 4(c), we observe that FI at $(70, 30)$ is not optimal and reaches an optimal fairness around $(100, 0)$. But at $(100, 0)$, the uplink utilization and mean download time both are poor. These results leave us with an important question, can we achieve an optimal behavior in terms of all the three parameters simultaneously, i.e., do uplink throughput, mean download time and FI can all be optimized together ? A similar sentiment is shared by Fan et al. (4). They showed through their analytical model that current version of BT is only one of the possible scenarios in the whole spectrum of possible BT settings of fairness, uplink throughput and mean download time, and a system could be designed on exactly what we want to optimize but not all.

5.4 Overall Effect of Our Model

We showed the effect of each of our techniques individually in sections 5.1, 5.2, 5.3. Now, we apply of these strategies together and compare the result with the reference BT protocol. We assume 10% strategic nodes in 1024 node swarm for various torrent file sizes. We use $(70, 30)$ peering policy for sending peer list to the new incoming node. Figure 5 depicts the results after incorporating all our techniques.

The effect of anti-strategic policy and bandwidth clustering adds up to show a considerable difference in the download time difference between BT and our model. As mentioned earlier, uplink throughput and download time both improve as a result of both anti-strategic behavior and bandwidth clustering. A consistent difference of around 4 to 5% in uplink utilization and difference of around 45 minutes in the mean download time is evident from figure 5(a) and 5(b). Both the policies,



(a) Download Time Difference vs File Size in MB (b) Uplink Utilization vs File Size in MB (c) Fairness Index vs File Size in MB

Figure 6 *Altruism effect on BitTorrent*

i.e. anti-strategic behavior and bandwidth clustering do not interfere with each other when used together. Bandwidth clustering is only used for issuing a peer list, while anti-strategic behavior mainly *TokenFromTracker* is built on top of the bandwidth stratum a node belongs to. Policies like Smart tracker and blacklisting are independent of bandwidth clustering and pose no interference to other policies of our model. Similarly, the FI is significantly superior in our model as compared to BT as shown in 5(c), signifying that in our model more nodes have share ratios around 1.0 and is comparatively fairer. Though, it is very difficult to achieve a perfect FI of 1.0, we present in section 5.5 that in some cases of non altruistic behavior, FI very close to 1.0 can be obtained but at the expense of mean download time and uplink throughput.

5.5 Effect of Altruism on the System

As mentioned earlier, one of our main finding is that altruism plays an important factor in improving the download time. There are many seeds who voluntarily offer their upload bandwidth in return for nothing, which in fact reduces many peer's download time. To quantify the effect of altruism, we use the altruism factor as described in section 5.5. We conducted three experiments: 1) Users are altruistic, i.e. they remain and offer their uplink bandwidth even after they have finished downloading. The *Altruism Factor* of these nodes is strictly greater than 0. 2) Users are not altruistic but they offer uplink till they reach a share ratio of 1.0, i.e. the *Altruism Factor* of these nodes is 0. 3) Users are greedy and leave the swarm as soon as they finish the downloading. The $Altruism factor$ of such nodes is strictly less than 0. We investigate the trends obtained in the main metrics when we vary our system to these variants.

From figure 6(a), we can see the effect of altruism in minimizing the mean download time. High capacity peers when engaged in voluntary uploading the content enhances the swarm mean download time. For a large file of 8192 MB, with 1024 nodes in the swarm, the difference in mean download time between the two schemes wherein peers stay in the swarm and where peers depart immediately after download is close to one hour. This result is quite intuitive as seeds offer many more copies to the swarm taking the load off the leeches, and they can utilize their uplink in obtaining other useful content from other peers, thereby minimizing their download time. The content uploaded by the seed is very important in this context.

On the other hand if nodes leave the swarm as soon as they reach a share ratio of 1.0 or immediately after their download, it will be up to the remaining nodes to pool in the uplink resources for all the nodes, thereby increasing the download time. These results reaffirm the notion that altruism is a very important factor in BitTorrent and one of the most important reasons for its enhanced performance. Similar results can be understood for uplink utilization. When seeds stay longer in the swarm and offer their uplink to fully utilize their outgoing bandwidth, the leeches use their uplink bandwidth with other leeches, thereby optimizing the uplink throughput. Same is not true when peers depart after downloading as the remaining peers have to search more for useful content and sometimes remain idle for lack of useful content. This reduces the uplink throughput.

Altruism, although, improves uplink throughput and mean download time, it is not fair to the contributing nodes. The FI is poor in the case when peers stay in the swarm after finishing their own download as the share ratio of seeds exceeds far above 1.0 and many leeches do not even have to upload a copy back to the swarm, i.e. their share ratio is much less than 1.0, creating this unfairness. In the case when peer stay till they upload a copy back to the swarm, most nodes depart the swarm with share ratio 1.0, thereby making FI close to 1.0. In the case when peers depart the swarm soon after download, the disparity is even higher. Only the few altruistic seeds offer uploads to most leeches in the swarm and this increases the disparity and reduces the FI. As nodes leave the swarm, the nodes in the swarm have to search more for the useful content and sometimes remain idle for the lack of interesting data. This reduces the uplink throughput and increases mean download time, though has a very high FI.

It is extremely difficult to reach a standard set of arguments where the behavior of the system is optimal in all the main metrics. Therefore, we try to reach a common ground where mean download time and uplink utilization is near optimal and FI is approximately in the range 0.9-0.95 or better. We show that this indeed can be achieved using the proposed policies in the paper.

6 Related Work

There has been considerable work done ever since Cohen (6) first created BT. Many simulation and analytical based studies have been reported till date. Most simulation based studies have focused on BT performance at various setups. Izal et al. (9) focused on the tracker log obtained from the Redhat 9 Linux Distribution. Their work enumerated the basic properties of the torrent i.e. most clients after finishing the download tend to stay in the pool for another 6.5 hours because they need manual intervention to close the BT client and stop uploading. They also reported average upload speed achieved during the run of the torrent. They have seconded the claim by Cohen (6) that *tit-for-tat* policy is effective in BT and gives near optimal results. Pouwelse et al. (18) also performed a study on a 8 month log obtained from a real life tracker of more than two thousand global components. Their main finding is that within p2p systems a tension exists between availability which is improved when there are no global components, and data integrity, which benefits from centralization. Sherwood et al. (21) have explained the *Slurpie* system which is very similar to BT. It uses an available bandwidth

estimation technique. All nodes downloading the same file contact the topology server. Using the information returned by the topology server the nodes form a mesh and propagates progress updates to other nodes. Slurpie protocol has been implemented and is available for download. Qiu et al. (20) modelled BT using fluid flow and conducted an analytical performance study. They have derived expressions for average number of seeds, leeches and download time using the node arrival and departure rate. They have shown that BT is scalable and performance improves as there are more users in the system.

Bharambe et al. (3) created a discrete event simulator to test BT on various different parameters. They showed the presence of significant altruism and unfairness in the swarm. They proposed to use TFT at the block level rather than rate based TFT to overcome unfairness. Fan et al. (4) in their analytical study showed that BT could be designed in several different ways, where achieving fairness among end users could be one end of the spectrum and minimizing the mean download time the other end. Legout et al. (11) showed that BT's piece replication using rarest first algorithm is efficient and to replace such a policy is not justified in the context of P2P file replication in the internet. They also showed that the newly incorporated choke algorithms in BT induces reciprocation and is robust to free riders. They also showed that choke algorithms is fair and better than bit level TFT.

Shneidman et al. (1) showed that BT indeed can be exploited using Sybil attacks (7) and by uploading garbage content. Other vulnerabilities and strategic attacks on BT have been mentioned in section 2.1.

We present the first and foremost work to defend BT against strategic attacks, not previously demonstrated. We perform the study of BT using the proposed anti-strategic policies and come to the conclusion that we can indeed have a fairer and more efficient swarms in terms of optimal mean download time and uplink utilization. The other part of our work validates the improved performance of BT while using bandwidth peer matching policy. We validated this concept using different set of experiments under different settings. Finally, we second the claim of Piatek et al. (17) that altruism is very important for improving overall swarm performance.

7 Summary and Conclusion

We presented some defense mechanisms and policies against the strategic BitTorrent clients. In particular, we showed that by using our proposed anti-strategic policies and bandwidth clustering, not only can the system be prevented from such cheating and strategic attacks but also overall system performance in terms of mean download time, uplink utilization and fairness can be improved. Our simulation results corroborate with the proposed theory. Clustering peers of similar bandwidth has shown to be very effective in utilizing the uplink capacity, and it reduces the mean download time. Anti-Strategic policies do not let cheating clients to stay longer in the swarm. They either are kicked off the swarm or they turn altruistic (from selfish), and the uplink resources of such nodes is utilized and is extremely important for the swarm. We believe our results can provide research insights for the development of new defence mechanism in present day BitTorrent clients to guard against the strategic attacks. Moreover, bandwidth clustering of similar nodes can

be easily incorporated into the clients straight away as it can be done with very minor protocol changes.

References

[Azu] Azureus. http://azureus.sourceforge.net/.

[1] (2004). *Faithfulness in internet algorithms*, Portland, OR, USA. ACM SIGCOMM.

[2] Adar, E. and Huberman, B. (2000). Free riding on gnutella.

[3] Bharambe, A. R., Herley, C., and Padmanabhan, V. N. (2006). Analyzing and improving a bittorrent networks performance mechanisms. In *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–12.

[4] Bin Fan, Dah-Ming Chiu, J. C. L. (2006). The delicate tradeoffs in bittorrent-like file sharing protocol design. In *14th IEEE International Conference on Network Protocols*.

[5] Choi, S. and Kim, C. (2001). Loss recovery time of impatient variant of tcp newreno.

[6] Cohen, B. (2003). Incentives build robustness in bittorrent. In *P2P Economics Workshop*.

[7] Douceur, J. R. (2002). The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK. Springer-Verlag.

[8] Hales, D. and Simon, P. (2005). How to cheat bittorrent and why nobody does. Technical report, Department of Computer Science, University of Bologna.

[9] Izal, M., Urvoy-Keller, G., Biersack, E. W., Felber, P. A., Hamra, A. A., and Garcs-Erice, L. (2004). Dissecting bittorrent: Five months in a torrent's lifetime. volume 3015 / 2004, pages 1–11. Springer Berlin / Heidelberg.

[10] Jain, R., Chiu, D., and Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301, Digital Equipment Corporation, Maynard, MA, USA.

[11] Legout, A., Urvoy-Keller, G., and Michiardi, P. (2006). Rarest first and choke algorithms are enough. In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pages 203–216, New York, NY, USA. ACM Press.

[12] Liogkas, N., Nelson, R., Kohler, E., and Zhang, L. (2006). Exploiting bittorrent for fun (but not profit). In *5th International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, CA.

[13] Locher, T., Moor, P., Schmid, S., and Wattenhofer, R. (2006). Free riding in bittorrent is cheap. In *5th Workshop on Hot Topics in Networks (HotNets), Irvine, California, USA*.

[14] Magharei, N., , Stutzbach, D., and Rejaie, R. (2005). Peer-to-peer receiver-driven mesh-based streaming.

[15] Medina, A., Lakhina, A., Matta, I., and Byers, J. (2001). Brite: an approach to universal topology generation. pages 346–353.

[16] Morris, R. (1997). Tcp behavior with many flows. In *ICNP '97: Proceedings of the 1997 International Conference on Network Protocols (ICNP '97)*, page 205, Washington, DC, USA. IEEE Computer Society.

[17] Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2007). Do incentives build robustness in bittorrent? In *NSDI'07*.

[18] Pouwelse, J. A., Garbacki, P., Epema, D. H. J., and Sips, H. J. (2005). The bittorrent p2p file-sharing system: Measurements and analysis. In *4th International Workshop on Peer-to-Peer Systems (IPTPS)*.

[19] Purandare, D. and Guha, R. K. (2006). Preferential and strata based p2p model: Selfishness to altruism and fairness. In *ICPADS (1)*, pages 561–570.

[20] Qiu, D. and Srikant, R. (2004). Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 367–378, New York, NY, USA. ACM Press.

[21] Sherwood, R., Braud, R., and Bhattacharjee, B. (2004). Slurpie: A cooperative bulk data transfer protocol.

[22] Small, T., Liang, B., and Li, B. (2006). Scaling laws and tradeoffs in peer-to-peer live multimedia streaming. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 539–548, New York, NY, USA. ACM Press.