

# Preferential and Strata based P2P Model: Selfishness to Altruism and Fairness

Darshan Purandare                      Ratan Guha  
School of Electrical Engineering and Computer Science  
University of Central Florida, Orlando, FL 32816 USA  
{pdarshan, guha}@cs.ucf.edu

## Abstract

*Peer to Peer models are based on user altruism, where a user shares his content. Real life p2p systems suffer from free rider's problem. Many nodes contribute much more than they should while others get a free ride for downloading the content. We present a simulation based study of BitTorrent like p2p systems. We propose a self punishing and self healing model that dissuades the users from cheating and encourages them to cooperate. A preferential based scheme is used to group nodes in the same stratum. This protocol uses publish based model and public key cryptography that prevents the nodes from cheating and enforces them to cooperate. Our results show that the node behavior changes from selfish to altruistic. Our proposed model ensures fairness, performs better in link utilization and minimizes the mean download time.*

## 1. Introduction

Last few years have seen the upsurge of third generation p2p models. Bit Torrent (BT) [1, 2, 3] is one of the very popular p2p systems used for bulk file download. Results [4] show that BT scales very well to large number of users and achieves near optimal performance in terms of uplink utilization, mean download time and fairness. The BT like p2p model is based on tit-for-tat policy, however, it lacks in terms of fairness. Some nodes end up uploading more than 6 copies while others almost get a free ride [5]. Our model aims to achieve the following.

- 1.Survivability: At all times, every block of the file exists in the swarm ensuring system survivability.
- 2.Minimum Mean Download Time: The model aims to minimize the average of the download times of all the nodes in the system.
- 3.Link utilization: The model aims to maximize total uplink utilization of nodes in the swarm.
- 4.Fairness: No node is forced to upload more than it has downloaded.

However, there is always a trade off involved when we try to accomplish these goals. We define stricter bounds to achieve good uplink utilization and minimize the mean download time, which in turn ensures fairness. We propose a preferential and strata based pairing scheme among the nodes. The idea is to group nodes with similar bandwidths together in a single stratum. The tracker issues every node a list of its preferred peers based on their respective stratum. Later on we show that this scheme of assigning the peers indeed is better than random selection of peers as in conventional BT model. We aim to achieve optimal system performance in terms of fairness, uplink utilization and mean download time. The emphasis is overall system benefit and not individual node gain. However, if free riders [5] exist in the system, it in turn affects overall system performance. As a result the contributing nodes suffer in terms of fairness. We propose the use of public key cryptography to alleviate this issue and prevent nodes from cheating.

Our main contributions are a preferential and strata based clustering scheme to group nodes and use of public key cryptography to prevent the nodes from cheating. We introduce the notion of TokenFromTracker and Published Upload Speed for the same. We have proposed a self healing and a self punishing model to dissuade selfishness and propagate altruism.

Section 2 gives an overview of BT like p2p model followed by related work in section 3. We present our model in Section 4. Section 5 describes the simulation setup with analytical study and simulation based results in section 6. Subsequently, we have our conclusion and references.

## 2. BitTorrent like P2P Models

BitTorrent (BT) [1, 2, 3] is a p2p application used for bulk file download. Conventional p2p systems were used for small sized data files with one to one connections possible between them. Large multimedia files and software distributions demand a fast and efficient mode of transfer. It exploits the uplink speed of end users while they are

downloading parts of the file. With BT, files are broken into small chunks typically 256 Kb. As the fragments are distributed to the peers in a random order, they can be reassembled on a requesting machine. Each peer takes advantage of the best connections to the missing pieces while providing an upload connection to the pieces it has already downloaded. This scheme has proven particularly useful in trading large files such as video, games and software source code. In conventional downloading, high demand leads to bottlenecks as demand surges for bandwidth from the host server. With BT, high demand can actually increase throughput as more bandwidth and additional seeds of the completed file become available to the group. Cohen [1] claims that for very popular files, BT supports about a thousand times as many downloads as HTTP and prevents server crashes evident in the HTTP downloads.

There are two types of nodes in the system. The nodes that have finished downloading the file and willingly offer uploads to other users are called as seeds. The nodes still in the process of downloading file are called as leeches. Leeches also offer uploads to other users as they download.

To share a file using BT, a user 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 joins the system he contacts the tracker to obtain a list of 40 peers including seeders and leeches who are in the swarm. A new node upon receipt of peer list contacts these nodes to obtain the file blocks. If the peer list goes below 20 due to departure of some nodes the new node makes another request to tracker to give him the addresses of some more peers.

Every node downloads as many blocks and as fast as they can. For each available source, the node considers the blocks of file available and then requests the rarest block among the peers. This is called as Local Rarest First(LRF) policy. The least replicated 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 looks for someone to upload the block.

A tit-for-tat policy is enforced to make sure that leeches do not get a free ride and also give back to the system by performing uploads. BT gives the best download performance to the nodes with maximum upload, a property known as "leech resistance". It discourages leeches from downloading the file without uploading it to anyone. This

policy forces everybody to contribute to the system to get maximum system benefit. Every node tries to limit the number of uploads at one instant to some small number say 5 to avoid having lots of competing TCP connections [4].

A technique called choking is used to limit the number of uploads. A node uses choking to block the upload connections to maintain its own performance. In general, the set of neighbors that a node is uploading to may differ from the set of nodes it is downloading from. Time to time every node performs optimistic unchoking which helps new user to get started. When a new user joins the system there is no way he can start downloading the blocks unless there is optimistic unchoking. The scheme helps the node realize if there is any other node giving better upload speed to him so that it can choke some other connection and unchoke him. Though BT is a good protocol for a broadband user, it is less effective for dial up connections, where disconnections are common. On the other hand, many HTTP servers drop connections over several hours, while many torrents exist long enough to complete a multi-day download.

### 3. Related Work

There has been considerable work done ever since Cohen [1] first came with the idea of BT. Many simulation and analytical based studies have been reported till date. Most simulation based studies focused on BT performance at various setups. Izal et al. [6] 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 [1] that tit-for-tat policy is effective in BT and gives good results. Pouwelse et al. [7] 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. [8] 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. Shrivastava et al. [9] have presented an incentive based streaming in p2p environment. Qui et al. [10] modelled BT using fluid flow and conducted an analytical performance study. They have derived expressions for average number of seeds, leeches

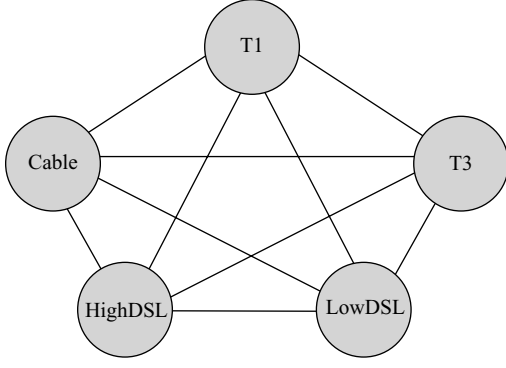


Figure 1. A Strata based mechanism

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.

Our work mainly focuses on classifying nodes in various strata so that peers mostly exchange packets with peers in the same bandwidth stratum. This is very crucial for good uplink utilization. On the fairness front, we employ a published based model where a node publishes his standard upload speed. To stop cheating, tracker creates a token called “TokenFromTracker” which uses public key cryptography to encrypt the token.

#### 4. Preferential and Strata Based Model

We propose a novel scheme to classify and group nodes in a similar bandwidth range into respective stratum. Nodes within different strata have the flexibility of communicating with each other for the requested file block. However, the nodes that lie within the same stratum are the primary preferred peers for file exchange followed by the nodes in the nearby stratum in terms of their bandwidth difference

In real life scenario we have nodes with heterogeneous bandwidths such as T3, T1, Cable, High DSL and Low DSL. We assume five bandwidth classes related as:

$$\Delta_1 > \Delta_2 > \Delta_3 > \Delta_4 > \Delta_5$$

Let,

$$\Delta_{(i,j)} = |\Delta_i - \Delta_j|$$

where  $\Delta_i$  and  $\Delta_j$  are the bandwidths of strata  $i$  and  $j$ . Therefore,

$$\Delta_{(i,j)} \leq \Delta_{(i,k)} \quad \text{for } |i - j| \leq |i - k|$$

where

$$i \leq j \leq k \text{ or } i \geq j \geq k$$

In our model, a node with bandwidth  $\Delta_2$  is most likely to interact with nodes from  $\Delta_2$  as  $|i - j|$  is 0. A second preference is given to  $\Delta_3$  and  $\Delta_1$  bandwidth nodes, which are the strata with the second minimum bandwidth difference.

$$\Delta_{(2,2)} < \Delta_{(2,3)}$$

$$\Delta_{(2,2)} < \Delta_{(2,1)}$$

Some fraction of the peer list will include nodes from remote strata. For example, in our case a node with  $\Delta_1$  speed would also have  $\Delta_3$ ,  $\Delta_4$  and  $\Delta_5$  bandwidth nodes as its peers. In our simulation we consider 60% peers from the same stratum, 15% each from the neighboring strata and 10% from remote strata. It can be denoted as (60, 15, 10). We also consider other possible distributions like (70, 10, 10) and (50, 20, 10). 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 is found to be among 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.

Generally such P2P systems comprise of large number of nodes. Nodes can enter and depart at any point of time. Moreover, the node behavior cannot be trusted. There is always a possibility of free riders in the swarm who selfishly download the content without contributing fairly to the swarm. We emphasize on the overall system performance and not on individual gains. Consequently the contributing nodes suffer on account of these free riders. This also has significant impact on the uplink utilization and fairness ensured by the preferential and strata based scheme. We propose a self healing and self punishing model to counter this issue.

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 the same upload speed as published throughout the time it is in the swarm. Every node upon arrival contacts the tracker. It sends PUS it is going to offer to the peers. An end user (node) can configure the BT client based on his preferences. Even if a node has good uplink speed he would not want to dedicate all his uplink speed. However, if the node publishes a lower uplink speed he will be placed in the lower stratum. So the node resorts to cheating by initially offering high PUS and later configuring his BT client to downgrade the uplink speed. Open source programs for BitTorrent gives an end user a chance to modify the protocol in the code. New programs like Azureus [11] 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 his 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.

We propose to create a token called "TokenFromTracker". This token is encrypted with tracker's private key. The payload contains Node ID, PUS and Arrival Time:

$$\{[\text{Node ID, Published Upload Speed, Time}]K_R\}K_U$$

For example, node A and B agree to communicate. They exchange their tokens and get the information of peer's PUS and other details by decrypting the token with tracker's public key. While the session is on, the peer can gauge the nodes offered upload speed. A node is immediately caught if it keeps upload bandwidth low for a certain period of time interval. The model associates some tolerance T with the upload speed i.e. if a node publishes an upload speed of U, then its uplink speed in the range U-T to U are all acceptable. If it falls below U-T the other node would wait for a small time interval t and eventually disconnect. Peer notifies the tracker about the cheating node. Tracker issues a warning to the cheating node. If number of complaints exceed the threshold K, the tracker brands him as a bad node and throws him out of the swarm. This is a consequence of our self punishing policy. Nevertheless, the warned users who cooperate with the protocol are allowed to stay in the swarm. This is in accordance with our self healing policy. Thus, our model is a self healing and self punishing one and turns the node behavior from selfish to altruistic enhancing the overall system performance.

After acquiring the peer list the node contacts the peers for the first file block. Once it procures the first file block it starts uploading as well downloading other file blocks. For the subsequent blocks Local rarest first (LRF) policy is enforced as in the conventional BT protocol [1]. The nodes continue to exchange the file blocks until they finish their respective downloads. Nodes may volunteer to stay in the swarm or leave. Nodes intimate the tracker while leaving the swarm.

## 5. Simulation Setup

We present the details of the simulation setup for our proposed preferential and strata based model. We present an evaluative comparison against the BT protocol. We have mainly focused on average time to finish the download, fairness in terms of the volume of content served and uplink bandwidth utilization. Under the assumption that ( $downlinkspeed > uplinkspeed$ ), 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 variance of share ratios and compare with BT to identify 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 implemented a discrete event custom simulator in Java. As mentioned in [4] network propagation delay is relevant only in the case of small sized packets such as request packets. Most P2P traffic is the data payload and ignoring propagation delay does not have a significant impact on the simulation results. For experimental purposes we make an implicit assumption about the network propagation delay and do not model it in our simulation. We do not model the TCP congestion and delays within the network. The bottlenecks are assumed to be either the uplink or downlink bandwidth and not any other point in the network. We believe that bulk file transfers lasts over an extended period of time and ignoring TCP dynamics for small intervals do not affect the simulation results. Pouwelse [7] findings are that the real world torrent downloads do not necessarily follow any particular arrival pattern. The arrival pattern of nodes in the swarm is assumed to be under Poisson distribution which is closest to the real world compared to any other distribution.

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 2048 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. The various bandwidth strata [2] we have considered are (10000 Kbps, 5000Kbps), (8000 Kbps, 4000 Kbps), (3000 Kbps, 1000 Kbps), (1500 Kbps, 384 Kbps) and (784 Kbps, 128 Kbps), where the first member of the tuple is the max download speed and the second is the max upload speed. The distribution of nodes among various bandwidth strata is uniform. On an average, every stratum has around 20% of the total nodes in the swarm. An implicit assumption is that as nodes finish their downloads they leave the swarm. We have injected around 25% of users who stay in the swarm to help others finish their download. The above mentioned numbers have been obtained from real life torrent examples [6].

We evaluated our model for the three main metrics viz. (i) Average download time (ii) Uplink utilization (iii) Fairness in terms of share ratio of each node in the swarm. Mathematically these are denoted as follows.

1. 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.

2. Uplink 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. Share ratio =  $\frac{U_i}{D_i}$ ,  
where  $U_i$  and  $D_i$  denote the uploaded and downloaded contents for node  $i$ .

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.

## 6. Results and Discussion

Our main objectives were to evaluate and compare the mean download time, fairness and percentage uplink utilization. Figures 2 and 3 show the comparison of Mean Download time and Uplink utilization in our model versus the conventional BT. Figure 2 is plotted for file size 256 MB and for 1024 MB. The simulations were run for users ranging from 128 to 8192 and time is calculated in second. It is evident from both the figures that the mean download time in our model has been considerably minimized. Furthermore, with increasing number of nodes, the mean download time decreases in our model. Thus, our model is scalable.

Figure 3 represents the percentage utilization of uplink bandwidth for file sizes 256 MB and 1024 MB. For both the cases our model has better uplink utilization compared to the conventional BT. For file size of 256 MB the utilization factors for BT and our model are 86% and 91% respectively while for 1024 MB the utilization factors are 83% and 88% respectively. Our model has consistently scored over BT for all number of users for the both the cases.

We have quantified fairness in terms of Share Ratio. Share Ratio is the ratio of uploaded volume content to the downloaded volume content. Share ratio of 1 is considered healthy and optimal where a node downloads a copy of the file and also gives back to the system the downloaded copy. Typically, original seed ends up uploading number of copies and there are always some very reliable seeds (not original) which stay in the swarm for a while to help others complete their downloads. Such seeds have a very high share ratio typically more than 5. We believe that a node should not be forced to stay in the swarm to finish the download. If nodes stay in the swarm, their altruism

is welcome but we believe that user altruism should not be forced. In the figure 4 we have depicted the fairness in terms of share ratio. This simulation run was done for 512 nodes in the system. It is evident from figure 4 that the variance of share ratio in our model is less as compared to BT. The variance of our model is 0.115725 while BT has 0.156253 which proves that our model is more consistent than BT in terms of fairness. The fact that our model kicks out free riders helps improve fairness among user nodes.

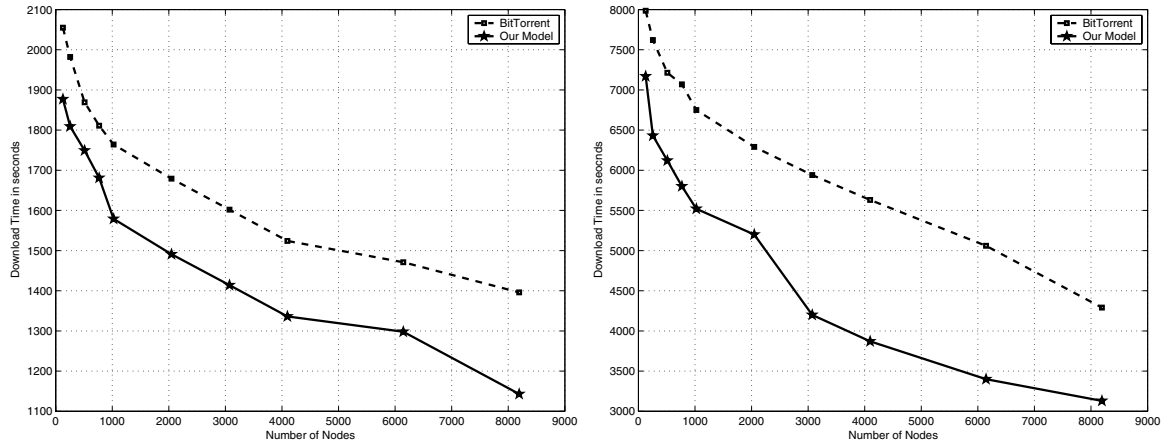
The results manifest that our model has achieved the above mentioned goals.

## 7. Analysis

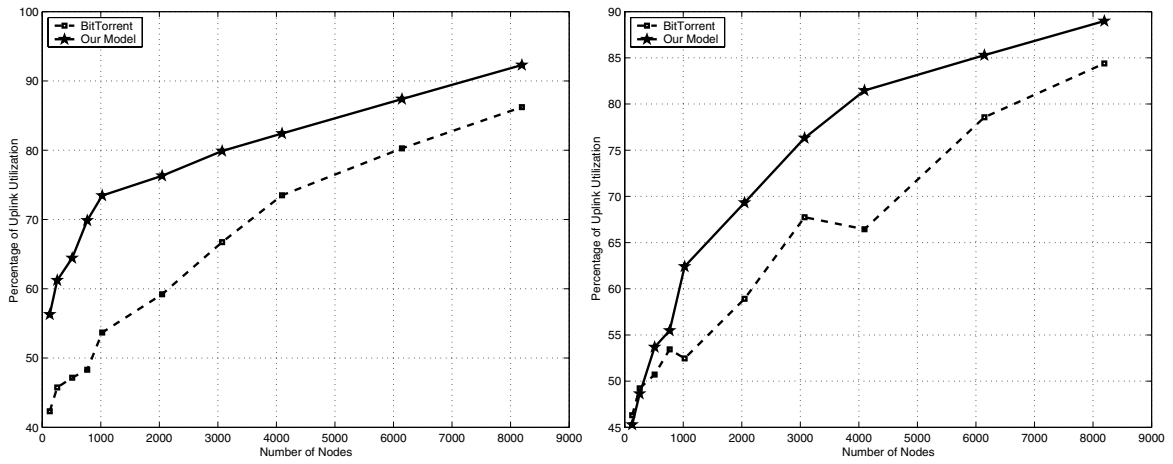
In sections 6 and 7 we compare our model to conventional BT and conduct an analysis of the same. We begin with quantifying the differences in the tracker overhead in our model and conventional BT. While in the conventional BT the peer list is assigned randomly, in our model the tracker finds best peers for every node. This task is computationally inexpensive as it has to search for peers in the stratum corresponding to the nodes published upload speed. The search space is considerably reduced. Tracker monitors the nodes in various strata and uses it to assign the peer list. Tracker creates a token called "TokenFromTracker" and encrypts with its private key. For instance if the Node Id is 64 bits, the published upload seed and miscellaneous information occupy another 64 bits, the tracker encrypts around 16 bytes for a single node. This encryption and computation can be done in parallel while computing the peer list. This does not incur any additional overhead. In addition to this the tracker also logs *bad node history* of the cheating nodes. Tracker warns a node if it gets complaints about him not adhering to the protocol. After monitoring that particular node, if the node behavior persists, the tracker may remove him from the swarm. Eventually the number of nodes the tracker has to monitor decrease. Above mentioned changes are not computationally expensive and do not flood tracker with lot of messages. They can be performed very well without any delay and degradation in the tracker performance.

We perform an analysis of the node behavior during our simulation. There are three stages in the run of the protocol shown in Figure 5.

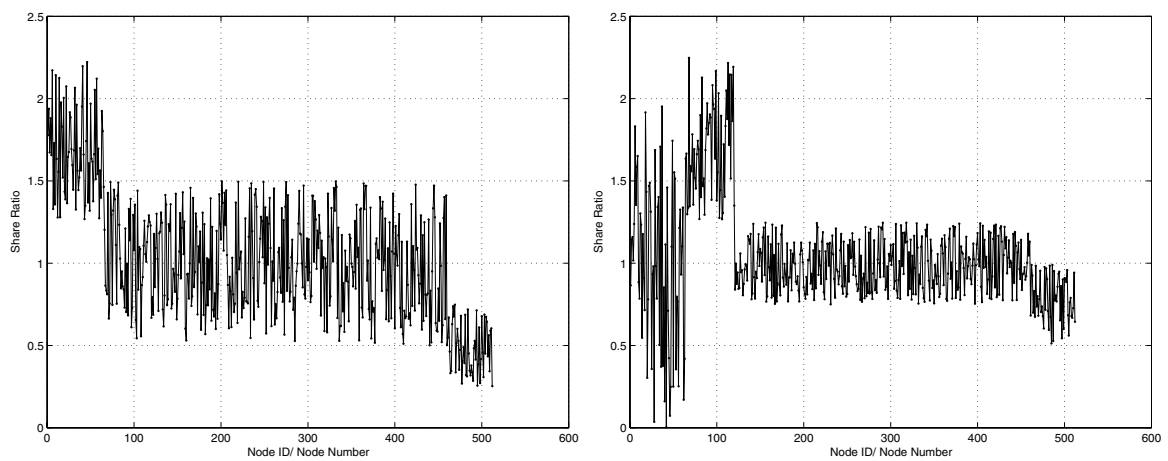
*Stage 1:* There are not enough number of users in the swarm. Due to less number of users in each stratum, the tracker cannot assign the peer list based on our protocol. Essentially, our model behaves the same as conventional BT in this phase for lack of nodes. This phase does not demonstrate any improvement in terms of uplink utilization as we cannot achieve preferential pairing of nodes. However, this phase does not dominate the total run time of the protocol and hence does not show a



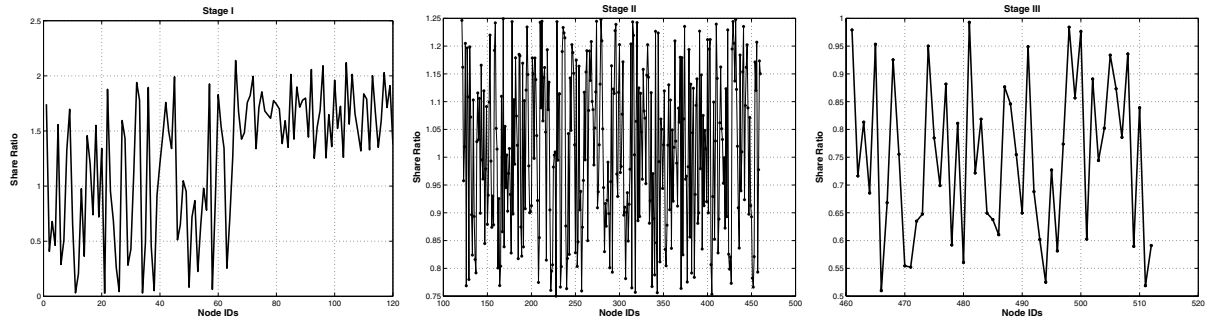
**Figure 2.** Comparison of Mean Download Time in Our Model versus BT for 256 MB and 1024 MB.



**Figure 3.** Comparison of Uplink Utilization in Our Model versus BT for 256 MB and 1024 MB.



**Figure 4.** Comparison of Share Ratios in Our Model versus BT for 512 users.



**Figure 5.** *Distribution of Nodes and Share Ratio in Stages I, II and III.*

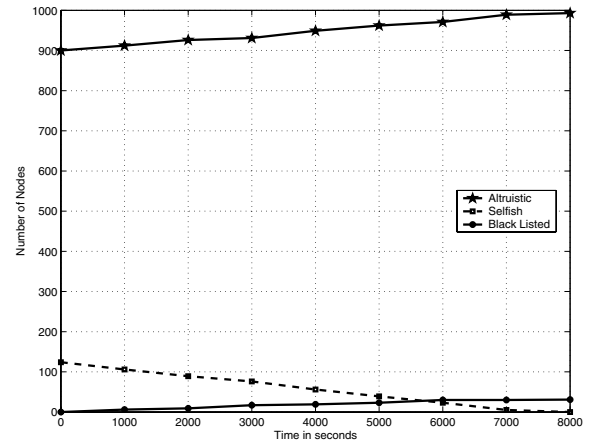
significant impact in the overall results. Figure 5 depicts this stage. It is evident that in this stage there is lot of disparity in the share ratio of the nodes.

*Stage 2:* There are considerable number of users present in the swarm. This state is called “Steady State”. This is the phase where our model is most dominant. The tracker allocates peer list based on preferential grouping in accordance with our model. As a result, good uplink utilization is achieved. Since nodes with similar bandwidths are involved in block exchange share ratio is close to 1. This ensures overall fairness in the system. This phase is marked by maximum transitions from selfish to altruistic nodes shown in Figure 6.

*Stage 3:* There is dearth of nodes again because the nodes that have finished the download leave the swarm. The seeds willingly offer uploads to help other users finish their downloads. That is why the share ratio of users involved in this phase is less than 1 and decreasing. It is evident that the share ratio steadily falls below 1.

Second phase dominates 90% of the protocol run time. Any improvement in this phase will be reflected in the overall results. First phase is similar to the conventional BT but its share is very less compared to the second and third phases that dominate the total run of the protocol. Results in the previous sections have demonstrated this fact. These changes are reflected in all three measures namely mean download time, fairness and uplink utilization.

In our simulation run, we injected around 10% cheating nodes. These nodes over a period of time turn altruistic during the simulation run. The nodes that cheat despite the warning are taken off the swarm by the tracker. In Figure 6, 900 nodes start as altruistic and 124 as selfish nodes. Towards the end 31 nodes are blacklisted and thrown out of the pool and rest 93 turn good. This shows that our model indeed turns user behavior from selfish to altruistic. In Figure 6 the number of altruistic nodes increase while cheating nodes decrease as they are thrown out of the swarm. This shows that our self healing and punishing policy holds good.



**Figure 6.** *Number of Selfish Users who turned Altruistic.*

## 8. Conclusions and Future Work

BitTorrent is inherently a very efficient protocol for bulk file transfer. But it does not achieve the best performance in terms of mean download time, fairness and uplink utilization. We present a refined model by adding strata for various bandwidth users for better pairing between the peers. Results show that this way of assigning peers is better than random selection of peers. Publish based model is an efficient way of classifying nodes in strata. Usage of public key cryptography adds flexibility and is a cost efficient solution to prevent cheating. Our analysis and simulation results have confirmed that our model is stable, scalable and performs well on all the three important metrics. Our self healing and self punishing policy helps turn user behavior from selfish to altruistic. Our results are promising and inspiring. Future goals related to this work are to analyze the graph theoretic properties such as node degree, max flow problem from node to sink of the BT like p2p system.

## 9. Acknowledgements

This work was partially supported by ARO under grant DAAD19-01-1-0502. The view and conclusion herein are those of authors and do not represent the official policies of the funding agency or the University of Central Florida, Orlando. We would like to sincerely thank Gautami Shirhatti for her invaluable comments and remarks which have helped to improve the content and presentation, and Cliff Czou for his participation in the discussion. Special thanks to anonymous reviewers whose suggestions and remarks have been very important for the paper.

## References

- [1] B. Cohen. *Incentives build robustness in BitTorrent*. P2P Economics Workshop 2003.
- [2] <http://www.bittorrent.com>
- [3] [http://en.wikipedia.org/Bit\\_Torrent](http://en.wikipedia.org/Bit_Torrent)
- [4] A. Bharambe, C. Herley, V. Padmanabhan. *Analyzing and Improving Bit Torrent Performance*. Microsoft Technical Report MSR-TR-2005-03.
- [5] E. Adar, B. Huberman. *Free riding on Gnetella*. Technical report, Xerox PARC, 2000.
- [6] M. Izal, G. Urvoy-Keller, E. Biersack, P.Felber, A. Al Hamara, L. Garces-Erice. *Dissection BitTorrent: Five months in a Torrents lifetime*. PAM, Apr 2004.
- [7] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. *A Measurement Study of the BitTorrent PeerToPeer FileSharing System*. Tech. Rep. PDS-2004-007, Delft University of Technology, Apr. 2004.
- [8] R. Sherwood, R. Braud, B. Bhattacharjee. *Slurpie: A Cooperative Bulk Data Transfer Protocol*. IEEE Infocom, March 2004.
- [9] V. Shrivastava, S. Banerjee. *Natural Selection in P2P Streaming: From the Cathedral to the Bazaar*. ACM NOSSDAV, Skamania, WA, June 2005.
- [10] D. Qiu and R. Srikant *Modelling and Performance Analysis of BitTorrent like Peer-to-Peer Networks*. SIGCOMM 2004.
- [11] <http://azureus.sourceforge.net/>