INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

# Optimal Stochastic and Adaptive Peer Position Update for Geographic Routing in a Network

**Thamayanthi M.**
Computer Science and Engineering
St. Joseph's College of Engineering and Technology, Thanjavur, India
**Sumathi M.**
Network Engineering, Kalasalingam University, Virudhunagar, India

***Abstract:***
*File sharing applications in peer-to-peer systems suffer from unfairness. Users use more bandwidth for downloading and leave the upload low or even zero there by contributing others to suffer. We present Fair Torrent, a new deficit-based distributed algorithm that accurately rewards peers in accordance with their contribution. We can implement Fair Torrent in a Bit Torrent client without modification to the Bit Torrent protocol and evaluate its performance against other widely used Bittorrent clients. Instead of downloading certain files from the nearer peer, it is recommended to download that file from the peer which has more upload than the nearer peer by finding its location.*

***Keywords****: Fair Torrent, Peer-to-Peer Networking, Service Quality, Bit torrent*

## 1. Introduction

Peer-to-peer file sharing is a popular, cheap, and effective way to distribute content. However, P2P file sharing applications suffer from a fundamental problem of unfairness. Many users free-ride by contributing little or no upload bandwidth while consuming much download bandwidth. By taking an unfair share of resources, free-riders cause slower download times for contributing peers. If a P2P system could guarantee fair bandwidth allocation, where by fairness we mean that a peer receives bandwidth equal to what it contributes, the system would be able to guarantee a certain level of performance for contributing peers. Many approaches have attempted to address the problem of fair bandwidth exchange. The most common approach is the tit-for-tat (TFT) heuristic employed by the popular file sharing system BitTorrent. TFT splits a peer's upload bandwidth equally among a subset of neighboring peers for a time duration called a round, then adjusts this subset each round based on estimates of their bandwidth rates from the previous round.
Fair bandwidth allocation in P2P systems can be difficult to achieve for several reasons. First, no central entity controls and arbitrates access to all resources, unlike scheduling fair allocation of bandwidth for a router . Second, the amount of bandwidth resources available is not known in advance, and peers cannot be relied upon to specify their own resources honestly. Finally, strategic peers and free-riders may try to take advantage of the system by contributing little or no bandwidth while consuming others' resources. We present FairTorrent, a new deficit-based distributed P2P algorithm that solves the problem of fair bandwidth exchange in the presence of free-riders and strategic peers. FairTorrent accurately rewards peers in accordance with their contribution. It runs locally at each peer and maintains a deficit counter for each neighbor, which represents the difference between bytes sent and bytes received from that neighbor. When it is ready to upload a data block, it sends the block to the peer with the lowest deficit. Unlike other approaches, FairTorrent uses a completely different mechanism that does not require an estimate of neighboring peers' rate allocation. It allows a peer to maximize its upload capacity utilization. It avoids long peer discovery and reaches a fast rate convergence. It does not need to estimate and predict peers' allocations.

## 2. Background Work

BitTorrent employs a rate-based TFT heuristic to incentivize peers to upload and attempt to provide fair exchange of bandwidth between peers. Peers participating in the download of the same *target file* form a *swarm*. The target file is conceptually broken up into pieces, typically 256 kB. Peers tell one another which pieces of the target file they already have and request missing pieces from one another. Requests are typically made for 16-kB sub-pieces. Peers that already have the entire file are *seeds*. Peers that are still downloading pieces of the file are *leechers*. TFT is used in a swarm to enable fair bandwidth.

BitTorrent peers tend to exchange data with other peers with similar upload rates over a large file download .Under some bandwidth distributions, the system has been shown to eventually converge to a Nash equilibrium. However, there is no evidence that this behavior extends to shorter file downloads, dynamic environments, skewed distributions of users, or modified but compatible BitTorrent clients. In fact, several modified BitTorrent clients have been developed that exploit different strategies to achieve better performance at the expense of users running unmodified BitTorrent. FairTorrent's new deficit-based approach does not need to explicitly set upload rates and avoids the pitfalls associated with having to estimate peers' changing rate allocations, dealing with insufficient and *ad hoc* round durations, and needing to explore a peer's entire neighborhood.

The problem of fair bandwidth allocation has perhaps been most extensively studied in the context of scheduling packets through a route. Given a set of flows with associated service weights, the problem is how to schedule packets to allocate bandwidth in proportion to the respective weights. While there are some similarities between the packet scheduling problem and the problem that FairTorrent addresses, the key difference is that in the latter case, peers have no explicit or assigned weights. Weights in packet scheduling correspond roughly to upload rates in P2P systems, but these rates are not assigned or known in advance. The resulting challenge in P2Psystems is how to provide fair bandwidth exchange given that the upload rates are not known, change dynamically, and can be difficult to estimate. While deficits have been used for packet scheduling , FairTorrent uses a completely different algorithm to solve a fundamentally different distributed problem.

**3. Fairtorrent algorithm**

FairTorrent implements a fully distributed algorithm that provides fair data exchange, without any global allocation or management service beyond what is already provided by BitTorrent. For compatibility with BitTorrent, FairTorrent uses the same BitTorrent protocol, torrent files, and tracker service.

*3.1. Leecher Behavior*

We first describe the basic algorithm run by the leechers. Let $Sent_{ij}$ be the total number of bytes that a leecher $L_i$ has sent to a leecher $L_j$ , and $recv_{ij}$ be the total number of bytes that $L_i$ has received from $L_j$. Each $L_i$ maintains a *deficit variable* $_{DFij}$ for each $L_i$ , where $DF_{ij}=Send_{ij}-Recv_{ij}$.Thus, a positive (negative) deficit implies that $L_i$ uploaded more (fewer) bytes to than it downloaded from $L_j$. Initially, each $DF_{ij}=0$. The values $DF_{ij}$ are maintained in sorted order by $L_i$ in a list called Sortedpeerlist. When $L_i$ is ready to send the next data block, it chooses to send that block to the peer with the smallest $DF_{ij}$.

*3.2. Seed Behavior*

Since seeds in a swarm do not download from peers in that swarm, using deficits to allocate bandwidth from a seed among leechers is of limited utility. FairTorrent allocates seed bandwidth to be split evenly among leechers by simply sending blocks in a round-robin fashion.

**Procedure1**:RECVBLOCK(*peer j,data_block p*)
**if** *Is Leecher*(*j*)and*ValidBlock*(*p*) **then**
*Recv* $_{ij}$ *Recv*$_{ij}$ *+ size(p)*
Df*ij* DF*ij* *size(p)*
*SortedPeerList,ReInsert(j)*
**endif**

*3.3. Exchanging Data*

According to the BitTorrent protocol, Li must unchoke $L_j$ to upload to $L_j$ . An unchoke message lets $L_j$ know that it can send requests for data blocks until a choke message is received. Azureus and BitTorrent unchoke only a few peers at a time. The reason for this is that it forces higher-uploading peers to push at a higher rate to each unchoked peer. This method allows higher-uploading peers to measure each other's rates and discover one another in a swarm over time.

PropShare is also able to split its upload rate unevenly to reward its neighbors according to their contributions. However, there are two main distinguishing characteristics that allow FairTorrent to reach a much faster rate convergence than PropShare. First, FairTorrent does not rely on estimating its peers' rate allocations toward it, as it does not need to explicitly set upload rates. By sending a block to the peer with the smallest deficit, or to whom it owes the most data, FairTorrent implicitly sends blocks faster to a peer from which it observes a faster rate. Because rate allocations change dynamically, and PropShare can only accurately estimate rates of several peers in each round, it fails to create an up-to-date view of its neighbors' rates and ends up splitting its own rate incorrectly as a result.

**Procedure 2:** SENDBLOCK(*allowed_bytes* B)
*n* ← *0;sz* ← *Size(SortedPeerList)*
**while**(B > 0)and(*n* < *sz)* **do**
*j* ← *SortedPeerList[n]*
**while** !*(*HPRF*(j)* and CWT*(j))*
*n* ← *n+1;j* ←*SortedPeerList[n]*
**end while**

if($n < sz$) **then**
*use_bytes = max*(B,BYTES REMAINS TO SEND(*j*))
*bytes_written* ← *send(j,use_bytes)*
B ← B-*bytes_written*
**if** BYTESREMAINSTOSEND(j)==0 **then**
$sent_{ij}$ ← $sent_{ij}$ + *block_size;*
$DF_{ij}$ ← $Df_{ij}$ + *block_size;*
*SortedPeerList.Reinsert(j)*
**end if**
*n* ← *n+1;*
**end if**
**end while**

- ***Bootstrapping****:* When a new peer joins the system, it needs to obtain a block to start trading. The fact that FairTorrent seeds send blocks in a round-robin to their 50 leechers, rather than sending to only several leechers over a 10-s round as does BitTorrent, allows a new peer to obtain a block faster. It may be useful to extend FairTorrent to allow a seed to maintain even more than 50 connections. It may also be useful to allow a leecher to send a block to a new peer with a very small probability. These changes would increase the probability that a new peer obtains a block quickly.
- ***Mega-Capacity Nodes****:* The experiments in this paper focus on upload capacity distributions where no leecher has a capacity that exceeds the sum of the upload capacities of all of its neighboring leechers. We show a wide range of such upload capacity distributions in, where FairTorrent peers observe a higher degree of fairness and bandwidth utilization as compared to other networks.
- ***Data Availability****:* FairTorrent leverages BitTorrent's *rarest first* policy when requesting a block from a leecher or a seed. It always asks for a least commonly available block among its neighbors. This policy makes it highly likely that a peer always obtains a block missing among some of its neighbors, and thus, it is able to trade. The intuition is that once a peer obtains a rare block, it can send it to many neighbors in exchange for distinct blocks. Also, procedure SENDBLOCK does not require a peer to always have a block to send to each neighbor.

*3.4. Live Swarms*
Live swarms consisted of 40 popular live BitTorrent swarms that distributed large files of 1–10 GB to thousands of users. For each swarm, we had all clients join it at the same time, but configured our clients not to talk to one another. We capped the upload and download rates of each client at 300 and 600 kB/s, respectively.
To avoid overusing bandwidth and memory resources on PlanetLab and avoid potential copyright infringement issues with live swarms, we did not download entire files, but instead joined each swarm for 1500 s and measured the download rate obtained by the clients. For live swarms, we observed that for each type of client, performance was highly correlated with the number of configured connections. As a result, we ran two sets of tests, first configuring each client with a 500-connection limit, the default for PropShare and BitTyrant, and then configuring each client with 50 connections, the default for Azureus.

**4. Fairtorrent Properties**
FairTorrent guarantees a high degree of fairness, fast convergence between a leecher's upload rate and its download rate from other leechers, and resilience to strategic peers. runs in *O(logn).*
The theoretical analysis of these properties is beyond the scope of this paper, but we give a brief theoretical intuition behind these properties. In addition, FairTorrent incurs low overhead. The most expensive step in a FairTorrent implementation is the reinsertion of a peer inside the *SortedPeerList* .

*4.1. Fairness and Service Error*
Service error measures the difference between the optimal fair share and the actual service received by a schedulable entity. in the context of p2p, we use service error to measure the maximum difference between the number of bytes a peer has contributed and received from other leechers at any time during a download session.

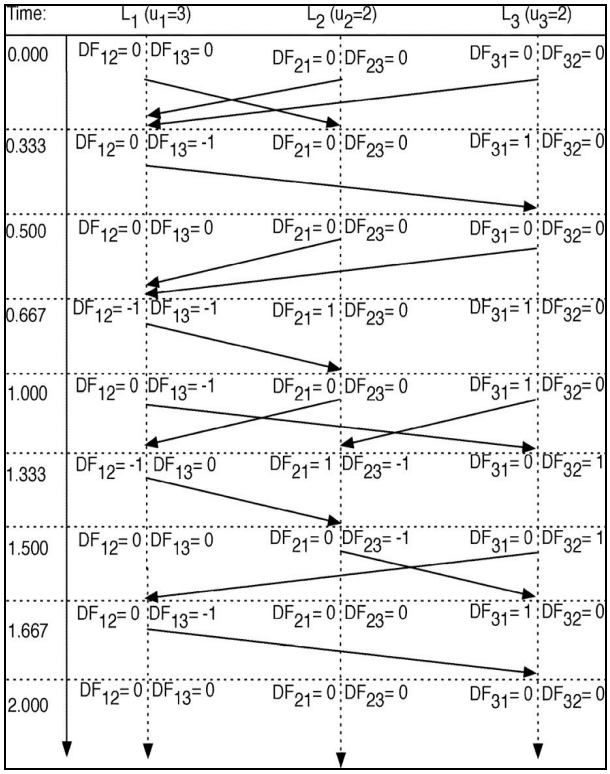


*Figure 1: FairTorrent algorithm for leechers $L_1$,$L_2$, and $L_3$ with upload capacities of 3, 2 and 2*

For an n-node network, for a wide range of upload rate distributions, and under a simplifying assumption that users always have data to exchange, we can prove that a peer in FairTorrent never incurs $E_{max}^{+}$ or $E_{max}^{-}$ of more than data blocks with high probability.

*4.2. Fast Convergence and High Utilization*

Fast convergence follows directly from the bounds on the service error. Since a leecher $L_i$ never contributes more than O(logn) blocks than what it receives, as soon as the peer sends a block to O(logn) peers, it will start to obtain full reciprocation. Thus, the maximum time to reach full reciprocation is the time it takes a peer to send O(logn) blocks.

One might think that, because of a strong correlation between upload capacity and completion times, the better performance for high-contributing FairTorrent leechers comes at the expense of the low-contributing ones.

*Whitewashing:* A *whitewasher* is a peer that free-rides, then leaves and reenters the system with a new identity and zero deficit to try to clear its debt. Whitewashing is a difficult problem for any P2P system.

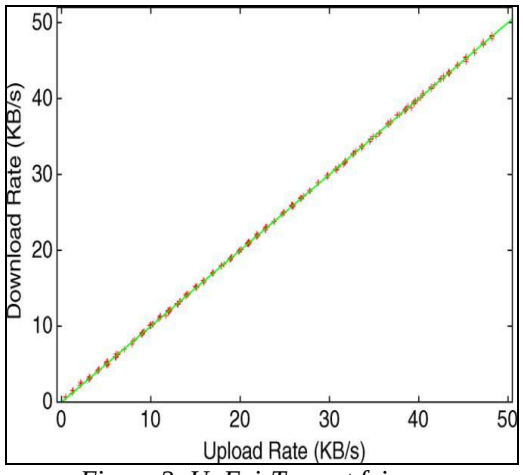


*Figure 2: U: FairTorrent fairness*

FairTorrent does offer some protection against whitewashing, as each time a free-riding peer enters the system, it will be limited in the number of free blocks that it can get from other leechers O (1), on expectation and O (logn) in the worst case for many capacity

distributions. One way to discourage multiple re-entries in FairTorrent would be for the seeds to allocate less bandwidth to new peers, making it uneconomical for the leechers to leave and reenter.

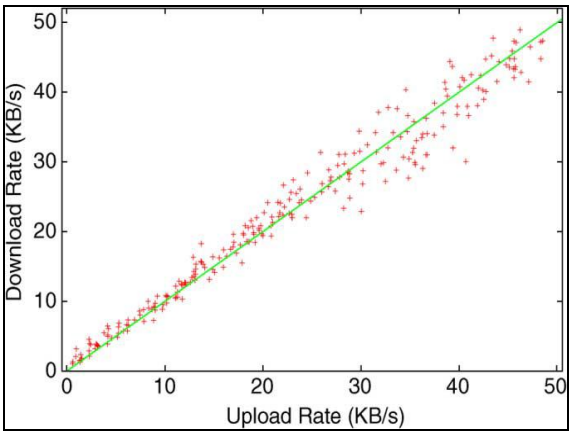


*Figure 3: U: Azureus fairness*

*4.3. Skewed Distribution*

To show what happens when a peer is surrounded by others that do not contribute much, we ran a single high uploader in a swarm of many low-contributing peers. We ran the same experiments with the same network of leechers and seeds as in Section V-A, but with a *skewed* distribution of upload capacities. The high uploader received an upload capacity of 50 kB/s, and the 49 low contributors received randomly selected upload capacities between 1 and 5 kB/s.

*4.4. Uniform Distribution*

The uniform distribution represents a wide range of peers with diverse upload capacities participating in a swarm. We used a network of 50 leechers and 10 seeds. A small seed-to-leecher ratio, 1:5, was used to show that the results do not depend on high data availability; the average ratio in live BitTorrent swarms is 1:1.All nodes were configured with download bandwidth capacities of 100 kB/s and upload bandwidth capacities of no more than 50 kB/s, reflecting a typical scenario where most ISPs allow users a download rate at least twice their allowed upload rate. Each leecher received an upload capacity randomly selected between 1 and 50 kB/s. Seeds were configured with upload bandwidth capacities of25 kB/s each, chosen to match the average upload bandwidth capacity of a leecher.

*4.5. Dynamic Live Distribution*

The dynamic live distribution represents peers with upload capacities taken from live torrents, which asynchronously join and leave, do not seed upon completion if they are leechers, and participate, in a larger, nonmesh network. We used a larger network of 100 leechers and 20 seeds and configured the upload capacities of the leechers based on a distribution of the upload capacities in live BitTorrent networks.

**5. Conclusion and Future Work**

FairTorrent is a new fully distributed P2P algorithm that accurately provides each peer with fair service commensurate with its bandwidth contribution. FairTorrent's deficit-based algorithm avoids the pitfalls of previous approaches that suffer from slow peer discovery, inaccurate bandwidth estimates, bandwidth underutilization, and complex tuning of parameters. FairTorrent does not require bandwidth estimates, a centralized system, peer reputation, or third-party credit keeping services. We compared FairTorrent against BitTorrent, Azureus, PropShare, and BitTyrant. We have shown that due to its high degree of fairness, compared to other P2P systems, FairTorrent is able to provide much better performance for contributing peers in a number of scenarios: 31% 68% better performance in the uniform distribution, 3-5times improvement for a high uploader in a skewed distribution, 37%–56% better performance for high contributors in a dynamic scenario with live capacities, and 60%–100% better performance in live swarms. FairTorrent is resilient to free-riders, low contributors, and strategic peers in both FairTorrent and non-FairTorrent networks.

Previous approaches for P2P file sharing incur long peer discovery times and imprecise and slow rate convergence, making them poor candidates for supporting streaming. The fairness and fast rate convergence properties of FairTorrent enable a client that uploads at a rate above the playback rate to also download a stream at the playback rate even over short time intervals. Extending FairTorrent to support streaming is a promising direction for future work.

Instead of finding the nearer peer as their host, the downloading peer can use the peer which has the most uploaded speed thereby reducing the slowness and thus it provides a better upload to another peer.

**6. References**


1. A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J. Martin, and C. Porth, "BAR fault tolerance for cooperative services," in Proc. 20th ACM SOSP, Oct. 2005, pp. 45–58.
2. Vuze, "Azureus," 2010 [Online]. Available: http://www.azureus.com
3. J. C. R. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queuing," in Proc. 15th IEEE INFOCOM, Mar. 1996, pp. 120–128.
4. K. Berer and Z. Despotovic, "Managing trust in a peer-to-peer information system," in Proc. ACM CIKM, Nov. 2001, pp.310–317.
5. A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving a Bit Torrent networks performance mechanisms," in Proc. 25th IEEE INFOCOM, Apr. 2006, pp. 1–12.
6. B. Cohen, "Incentives build robustness in BitTorrent," presented at the 1st Workshop Econ. Peer-to-Peer Syst., Jun. 2003.
7. F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servants in a P2P network," in Proc. 11th WWW, May 2002, pp. 376–386.
8. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in Proc. ACM SIGCOMM, Sep. 1989, pp. 1–12.
9. eMule, "eMule," 2010 [Online]. Available: http://www.emuleproject. net
10. B. Fan, D. Chiu, and J. Lui, "The delicate tradeoffs in BitTorrent-like file sharing protocol design," in Proc. 14th IEEE ICNP, Nov. 2006, pp. 239–248.
11. S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in Proc. 13th IEEE INFOCOM, Jun. 1994, vol. 2, pp. 636–646.
12. M. Ham and G. Agha, "ARA: A robust audit to prevent free-riding in P2P networks," in Proc. 5th IEEE P2P, Aug. 2005, pp. 125–132.
13. S. Jun andM. Ahamad, "Incentives in BitTorrent induce free-riding," in Proc. 3rdWorkshop Econ. Peer-to-Peer Syst., Aug. 2005, pp. 116–121.
14. S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The Eigen Trust algorithm for reputation management in P2P networks," in Proc. 12th WWW, May 2003, pp. 640–651.
15. A. Legout, N. Liogkas, E. Kohler, and L. Zhnag, "Clustering and sharing incentives in BitTorrent systems," in Proc. ACM SIGMETRICS,Jun. 2007, pp. 301–312.
16. D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee, "BitTorrent is an auction: Analyzing and improving BitTorrent's incentives," in Proc. ACM SIGCOMM, Aug. 2008, pp. 243–254.
17. H. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robison, L. Alvisi, and M. Dahlin, "FlightPath: Obedience vs choice in cooperative services,"in Proc. 8th OSDI, Dec. 2008, pp. 355–368.
18. H.Li, A. Clement, E.Wong, J. Napper, I.Roy, L. Alvisi, and M. Dahlin, "BAR gossip," in Proc. 7th OSDI, Nov. 2006, pp. 191–204.
19. Q. Lian, Y. Peng, M. Yang, Z. Zhang, Y. Dai, and X. Li, "Robust incentives via multi-level tit-for-tat," presented at the 5th IPTPS, Feb.2006.
20. T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free riding in BitTorrent is cheap," presented at the 5th HotNets, Nov. 2006.
21. T. Ngan, A. Nandi, and A. Singh, "Fair bandwidth and storage-sharing in peer-to-peer networks," presented at the 1st IRIS Student Workshop, Aug. 2003.
22. T. Ngan, D. Wallach, and P. Druschel, "Enforcing fair sharing of peer-to-peer resources," in Proc. 2nd IPTPS, Feb. 2003, pp. 149159.
23. A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," IEEE/ACM Trans. Netw., vol. 1, no. 3, pp. 344