BitTorrent for the Less Privileged

Umair Waheed Khan and Umar Saif
LUMS Computer Science Department, Lahore, Pakistan
{umair.waheed, umar}@lums.edu.pk

ABSTRACT

BitTorrent is a hugely popular peer-to-peer file sharing system. In countries where broadband Internet is widespread, BitTorrent accounts for as much as 70% of the overall Internet traffic. In contrast, in developing countries, BitTorrent is almost unusable on the typically low bandwidth dialup connections.

In this paper, we present a BitTorrent client called BitMate that is designed to enhance the performance of hosts with low-bandwidth connections. Importantly, BitMate enhances the performance of low-bandwidth nodes without cheating, circumventing the fairness policy of BitTorrent or adversely affecting the performance of other peers. In fact, BitMate drives its performance by scrupulously implementing the fairness philosophy of BitTorrent.

BitMate outperforms vanilla BitTorrent by as much as 70% in download performance, while at the same time improving upload contribution by as much as 1000%! BitMate also outperforms strategic clients like BitTyrant in low-bandwidth conditions by as much as 60% in download performance.

Categories and Subject Descriptors

C.2.1 [Computer Communication Networks]: Network Architecture and Design

General Terms

Design, Experimentation, Measurement

1 INTRODUCTION

BitTorrent [5] is a hugely popular peer-to-peer file exchange protocol. In countries where broadband Internet is widespread, BitTorrent accounts for as much as 30-50% of the overall Internet traffic [1]. In contrast, in developing countries, BitTorrent is almost unusable on the typically low bandwidth dialup connections and accounts for less than 10% of the overall traffic [1].

At first, this may appear surprising since BitTorrent is often blamed for enhancing the performance of low-bandwidth clients at the expense of higher bandwidth nodes [2]. On the contrary, we found that BitTorrent has disproportionately poor performance for low-bandwidth peers. Figure 1 shows that the performance of a BitTorrent client deteriorates sharply, almost in a step-like function, as its bandwidth is reduced from 200 kilobytes/sec to 5 kilobytes/sec. If the bandwidth of a client is in the bottom percentile of a swarm, its download rate becomes almost negligible.

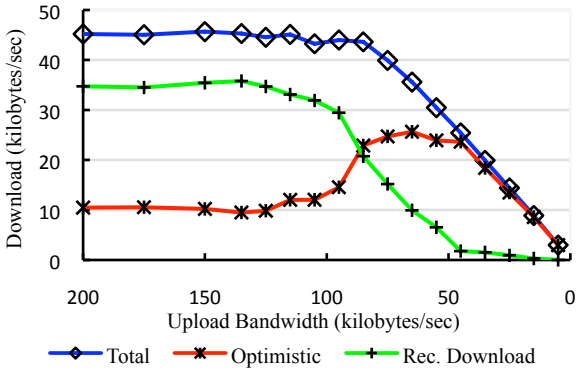

Figure 1: Performance of a BitTorrent client deteriorates as its bandwidth is reduced. As the bandwidth is reduced, *Total* downloaded data falls sharply, while more data is downloaded via *Optimistic* downloads instead of Reciprocal Downloads (*Rec. Download*) from other peers.

Ostensibly, a low-bandwidth peer does not perform well because it cannot download at a high bandwidth from other peers. An immediate temptation, therefore, is to somehow increase the bandwidth a low-bandwidth peer uses for download. One possible solution is to minimize the bandwidth a peer dedicates to uploading to other peers. Strategic clients like BitThief [3] and BitTyrant [2] achieve this by minimizing their upload contribution to other peers.

However, strategic clients not only contradict fairness in BitTorrent, surprisingly they also do not perform well in low-bandwidth conditions. In fact, we found that clients that do not upload at all, such as BitThief [3], perform even worse than vanilla BitTorrent in low-bandwidth conditions. In this paper, we advocate a diametrically opposite strategy for improving the performance of low-bandwidth clients. We show that in low-bandwidth conditions, an honest client, that strives to maximize its upload contribution, performs significantly better than both vanilla BitTorrent and strategic clients designed to minimize their upload contribution.

This paper makes three key contributions:

1. In contrast to a large body of recent work that argues that low-bandwidth clients benefit by an unfair strategic behavior that minimizes their upload contribution, we demonstrate that there is no incentive for low-bandwidth clients to cheat since they actually perform better by enhancing their upload contribution.
2. Our analysis shows that low-bandwidth clients fail to *fairly* utilize their download bandwidth even when there are other peers in the swarm that can offer them good download performance.
3. Based on our analysis, we present a BitTorrent client called BitMate that is designed to enhance the performance of hosts with low-bandwidth connections by maximizing it upload

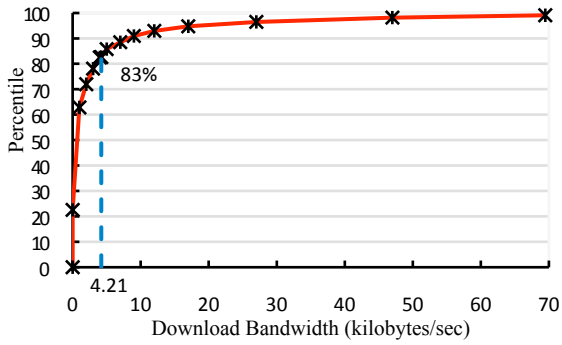

Figure 2: Average download bandwidth of peers in the Indian subcontinent.

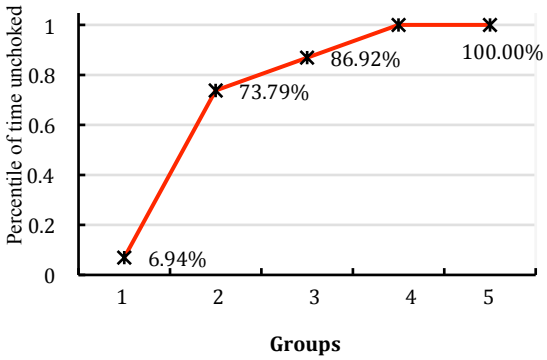

Figure 3: Percentile of time spent by a peer group in the unchokee list of other nodes in the swarm. Bandwidth ranges between 5 kilobytes/sec (group 1) to 200 kilobytes/sec (group 5).

contribution. Importantly, BitMate enhances the performance of low-bandwidth nodes without cheating, circumventing the fairness policy of BitTorrent or adversely affecting the performance of other peers. BitMate outperforms vanilla BitTorrent by as much as 70% in download performance, while at the same time improving upload contribution by as much as 1000%! BitMate also outperforms strategic clients like BitTyrant in low-bandwidth conditions by as much as 60% in download performance.

In this paper we present the first large-scale evaluation of BitTorrent in low-bandwidth conditions. As a point of reference to define a low-bandwidth client, we measured the average upload bandwidths of 6045 unique IPs from the Indian sub-continent using a modified BitTorrent client. The results of the experiments are shown in Figure 2, with the average upload bandwidth of a BitTorrent client in the Indian sub-continent to be 4.21 kilobytes/sec. With recent studies reporting the average bandwidth of a BitTorrent client to be close to 100 kilobytes/sec, clients in the developing-world invariably find themselves at the bottom of the pack in a swarm.

In the experimental evaluation presented in this paper, we use synthetic swarms using private torrents on the Planetlab testbed [4]. In our experiments, we used over 130 geographically diverse nodes on PlanetLab. All of our experiments were conducted over a period of 18 months, in the naturally diverse networks conditions of BitTorrent. Other than parameterizing our clients with our target upload and download bandwidths, we do not impose any restrictions on the naturally diverse conditions of Planetlab. For each experiment, we use between 55-130 randomly chosen nodes from Planetlab.

We have made our BitMate client available publicly, including source code. At the time of writing the paper, BitMate has been downloaded by more than 25,000 people from over 173 countries.

2 PLIGHT OF A LOW BANDWIDTH PEER

In the fair tit-for-tat world of BitTorrent [5], a low-bandwidth client does not have sufficient bandwidth to participate in the p2p system as a first-class citizen. With its paltry upload capacity, a low-bandwidth client rarely offers upload at a speed that earns it a right to download from another peer later. As a result, a low-bandwidth client is repeatedly snubbed by higher bandwidth peers when attempting to download a file and mostly relies on occasional benevolence (through optimistic unchoking) by other peers to make progress. Figure 1 shows the increasing percentage of data downloaded unfairly by a client as its bandwidth is decreased. As the bandwidth of a node is reduced, its primary mechanism for download shifts from (fair) reciprocal unchokes to gratuitous downloads due to optimistic unchokes from other peers.

Figure 3 illustrates this point further by plotting the percentage of times a client appears in the unchoke list of other peers in its swarm. In this experiment, we divide up the swarm in 5 groups of leechers with similar upload and download bandwidth, ranging between 5 kilobytes to 200 kilobytes. In our experiment, Group 1 nodes have the lowest bandwidth (5 kilobytes/sec), followed by Group 2 nodes (20 kilobytes/sec) with higher bandwidth, Group 3 with 100 kilobytes/sec, Group 4 with bandwidth of 150 kilobytes/sec and group 5 with 200 kilobytes/sec. The results show that the chances of a client earning a reciprocal unchoke by another peer falls sharply as its bandwidth is reduced within its swarm. Importantly, this sharp decline happens despite the composition of the swarm in our experiments: every group in the swarm (denoted on the X-axis) comprises an equal number of nodes with the same bandwidth, giving everyone an equal chance to form tit-for-tat relationships within its group. The reason for this sharp decline is captured in figure 4.

Figure 4 shows the number of *reciprocal* unchokes received by each group in our experiments. The tit-for-tat fairness policy of BitTorrent should cause peers to interact with others in their own group more frequently, resulting in tight clusters within each group. Earlier studies of BitTorrent have specifically highlighted the presence of such clusters [6]. However, in our experiments, we observe that while high-bandwidth nodes form mutually beneficial clusters, low-bandwidth clients do not receive any noticeable reciprocal unchokes, even from peers within their own group (*the results shown in figure 6 are normalized to offset the difference in bandwidth across peer groups; the normalization discounts the fact that high bandwidth clients naturally unchoke more peers).*

Figure 4 highlights the key result from our experiments: even when there are bandwidth-matched peers in a swarm, low-bandwidth nodes fail to establish mutually beneficial tit-for-tat relationships with them. Instead, they waste most of their goodwill in attempting to upload to and download from high-

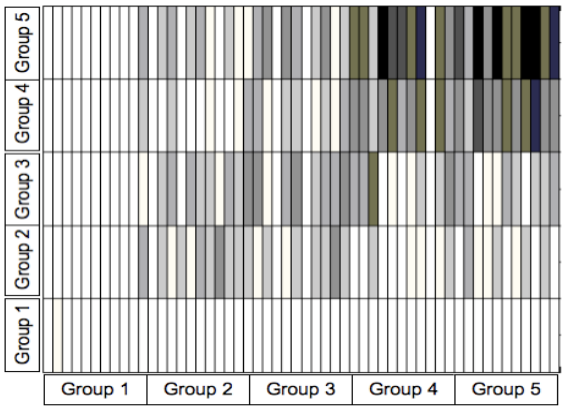

Figure 4: Number of times peers in each group unchoke each other, averaged over all runs. Darker regions represent higher number of mutual unchokes within a group. Low-bandwidth nodes (group 1, 2, 3) do not form mutually beneficial clusters.

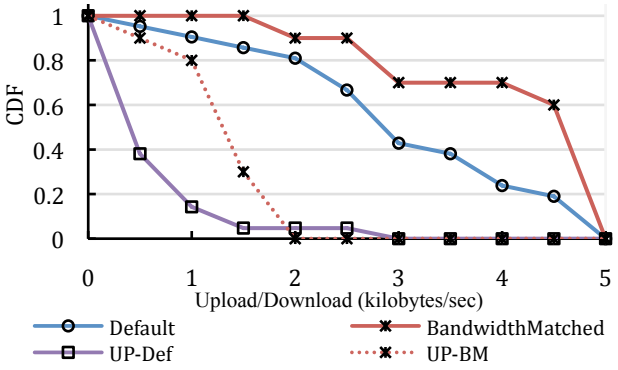

Figure 5: BitMate's ROU enhances both its download performance and upload contribution.

bandwidth peers, which do not reciprocate. This is because a BitTorrent client avoids low-bandwidth peers when downloading a piece to achieve good performance; in each round, a client unchokes the top N peers that have afforded it the best download bandwidth in the recent past. In other words, when given a choice of peers that have the piece a BitTorrent client needs, it always downloads from the peer that offers it the best download bandwidth. This greedy peer selection is designed to enable a BitTorrent client to efficiently utilize its download capacity.

However, as a flip-side of this greedy peer selection policy, most of the slots in the top N unchoke list of a low-bandwidth node go "wasted" since high-bandwidth unchoked peers do not receive an upload rate from it that merits it a place in their unchoke list. We call this *wasted goodwill*. Ironically, due to the wasted goodwill, low-bandwidth peers also overlook other low-bandwidth peers and hence fail to form mutually beneficial connections with anyone in the swarm (as shown in figure 4).

3 BITMATE: DESIGN AND EVALUATION

Based on our analysis of low-bandwidth peers in BitTorrnet, we designed BitMate, that achieves good performance by enabling better sharing between bandwidth-matched low-bandwidth peers.

A traditional BitTorrent client uses two principal mechanisms to establish peer-to-peer relationships with other clients: (1) Reciprocal unchokes, and (2) Optimistic unchokes. The former underpins the tit-for-tat fairness mechanism while the latter is used for exploration of the swarm to discover peers with which fair tit-for-tat relationships can be established. Previous work on BitTorrent [7][6][2] has focused on changing the tit-for-tat reciprocity of BitTorrent to either enhance its performance or make it more robust against free-riding clients. However, these approaches impact the fairness of BitTorrent and often mandate universal adoption by all the peers to be effective.

BitMate neither changes reciprocity nor mandates universal adoption to be effective. Instead, BitMate focuses on minimizing the wasted goodwill of low bandwidth clients. To accomplish this, BitMate modifies the optimistic unchoking policy of BitTorrent to make it more meaningful for low-bandwidth clients.

When unchoking a peer optimistically, a BitTorrent client is optimistic that the unchoked peer will select the client for downloading a piece and will in turn unchoke the client in reciprocation. Unfortunately, as explained before, optimistic unchokes of a low-bandwidth client towards a high-bandwidth peer mostly go in vain; due to its low upload bandwidth, an optimistic unchoke by a low-bandwidth node fails to earn it a place in the top N reciprocal unchoke list of a high bandwidth peer. Therefore, instead of wasting optimistic unchokes on high bandwidth peers, a BitMate client optimistically unchokes those peers that have a similar low-bandwidth. As a result, a BitMate client invests its scarce upload bandwidth on peers that are most likely to reciprocate. We call this realistically optimistic unchoking (ROU). At the same time, BitMate leaves the tit-for-tat reciprocal unchoke policy untouched to uphold the fairness of BitTorrent. This leads to both increased performance and fairness since low-bandwidth clients can quickly form mutually beneficial peer-to-peer connections.

Figure 5 gives the performance gain due to this scheme (BandwidthMatched), outperforming the default client by up to 30-40% times in download performance. Figure 5 also shows that BitMate uploads as much as twice compared to vanilla BitTorrent (UP-BM vs UP-Default).

The effectiveness of ROU depends on estimating the bandwidth of other peers to find low-bandwidth peers. BitMate uses a surprisingly simple mechanism to find other peers with matching bandwidth: instead of directly measuring and reporting bandwidth, a BitMate client simply matches the frequency of its own HAVE messages with other peers in its swarm. Clients with a similar frequency of HAVE messages, generated by a BitTorrent client to announce the download of a new piece, are good candidates for forming mutually beneficial peer-to-peer relationships.

3.1 Enhanced Clustering

ROU can improve performance if the low-bandwidth clients in a swarm possess pieces that are of mutual interest. If low-bandwidth nodes do not have mutually distinct pieces of a file,

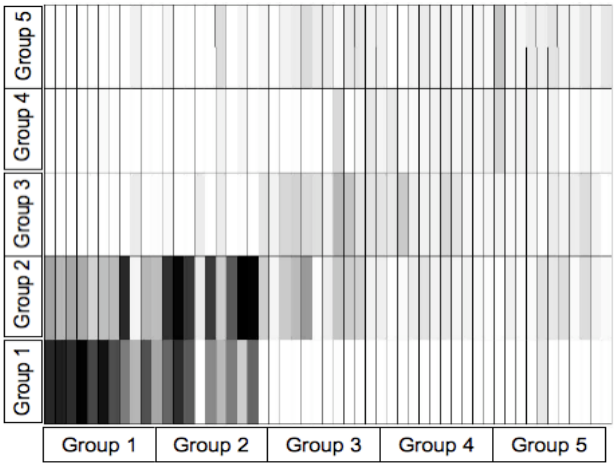

Figure 6: Low-bandwidth nodes (Groups 1, 2) cluster better due to ROU and Enhanced Clustering.

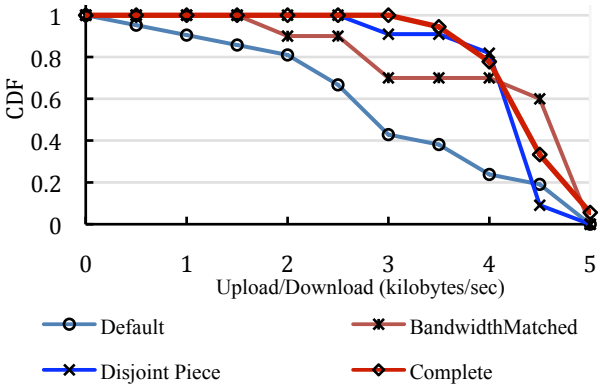

Figure 7: BitMate's download performance with ROU, Enhanced Clustering (Disjoint Piece) and Enhanced Sharing (Complete).

ROU's impact is limited. Unfortunately, due to their limited download bandwidth, low-bandwidth nodes generally have less data than higher bandwidth nodes in a swarm. Additionally, the rarest-block-first policy of BitTorrent often causes low-bandwidth peers to compete for the same (rarer) block stored at high-bandwidth peers.

Therefore, instead of competing with each other to download the blocks from high-bandwidth peers, BitMate clients download distinct blocks of a file from high bandwidth peers. This improves performance since low-bandwidth clients download from (reluctant) high-bandwidth peers only when necessary. This also improves fairness since it minimizes the data gratuitously downloaded by low-bandwidth clients from higher bandwidth peers -- instead encouraging mutually beneficial tit-for-tat connections between matching low-bandwidth peers.

BitMate does not introduce any additional messages to enable enhanced clustering between low-bandwidth clients. Instead, BitMate clients simply consult their log of HAVE messages, much like vanilla BitTorrent, before requesting a peer to download data. However, unlike a vanilla BitTorrent client which prefers to download a piece from the peer with the highest upload bandwidth, a BitMate client prefers to download from a bandwidth-matched peer if one has the required block. In essence, this causes BitMate clients to form a cluster, in which they try to make the best use of any opportunity to download from high-bandwidth peers (by downloading non-overlapping blocks) and then share this data over more stable tit-for-tat connections within the cluster.

Figure 6 illustrates the enhanced clustering between low-bandwidth BitMate with ROU and enhanced clustering optimizations (compare with figure 4). Enhanced data sharing accentuates the positive impact of ROU, resulting in better performance and fairness compared to vanilla BitTorrent. Figure 7 shows the improved download performance of BitMate after the enhanced clustering optimization (along with ROU), denoted as "disjoint piece".

3.2 Enhanced Sharing

ROU and enhanced clustering depend on the ability of a client to efficiently upload data. However, since BitTorrent clients exchange data at the granularity of a piece, typically between 256-512 kilobytes/sec, low-bandwidth clients often struggle to put together a piece that they can share with other peers. This is because a low-bandwidth client is frequently choked by other peers before it has a chance to complete the download of a piece. Due to the data discarded by these 'pre-mature' chokes, a client may end up downloading (at least some fraction of) the same piece multiple times. Such downloads, therefore, not only waste the download bandwidth of a choked node, they also exacerbate the degree of free-riding by low-bandwidth clients.

BitMate is designed to avoid redundant downloads that result due to the asynchronous architecture of BitTorrent. BitMate avoids redundant downloads by promptly sending a CANCEL request to a peer that chokes it. Traditionally, the CANCEL request is used only in the end game mode when a client aggressively requests a piece from all the peers in the swarm to avoid the last-block-problem. BitMate, on the other hand, promptly sends a CANCEL request as soon as a peer chokes it. This both conserves its download bandwidth, by avoiding redundant download, and improves fairness by preventing download from peers that have choked the client.

Still, even with the optimization to avoid redundant downloads, low-bandwidth nodes, due to their bandwidth, take a long time to string together an entire piece. This hampers the ability of a low-bandwidth client to share data. Reducing the size of the piece is not possible since piece size is dictated by the publisher and smaller piece size substantially increases the size of the *.torrent* metafile.

BitMate implements pipelined uploads to enable a low-bandwidth client to start sharing data without assembling a complete piece. In pipelined uploads, BitMate clients share data at the granularity of individual 'blocks', which make up a 'piece' in BitTorrent. The reason an individual block is not used as a unit of exchange in BitTorrent is because the integrity of an individual block cannot be verified in BitTorrent; integrity checks can only be performed at the level of a piece by

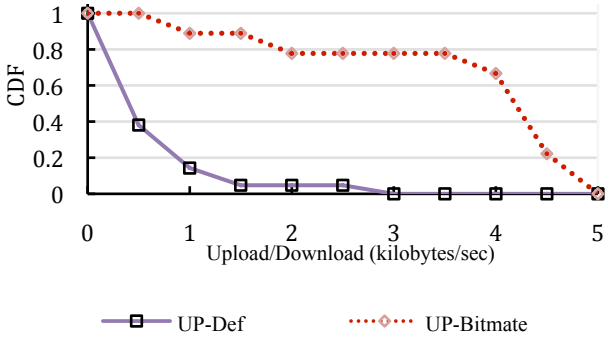

Figure 8: Comparison of upload contribution of BitMate (UP-Bitmate) and default BitTorrent.

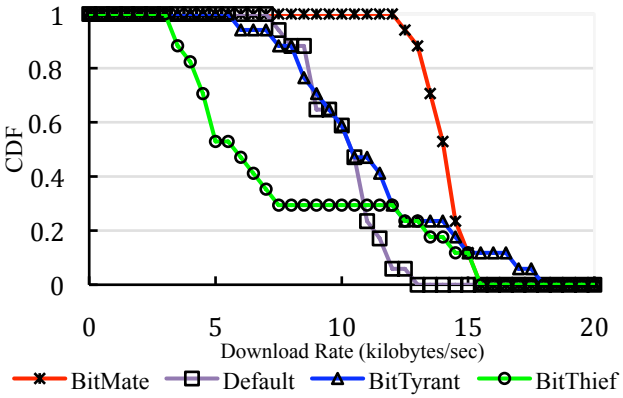

Figure 9: Comparison of BitMate with strategic clients with 20 kilobytes/sec (group 2).

comparing the checksum of the piece with the one stored in the .torrent file.

In order to aggressively upload blocks, BitMate clients start sharing blocks as soon as they download them (without waiting for a piece to be completed downloaded). To implement this, BitMate clients implement an additional message, HAVE_BLOCK, which they generate to announce the availability of a new block. This message is sent only to other BitMate peers having low bandwidth, to both avoid compatibility issues with other clients and to strengthen the clustering behavior of low-bandwidth BitMate clients. The performance and fairness gain due to aggressive block-level sharing easily offsets the rare chance of downloading a corrupted block. Still, BitMate clients, when publishing a file, also generate a Merkle Torrent, as proposed by BEP 30 [9], to enable flexible block-level sharing and integrity verification with other compatible clients. Traditional torrents are also generated to maintain compatibility with other clients. Figure 7 shows the improvement in the performance of low-bandwidth BitMate clients due to all three optimizations (denoted as 'complete'): ROU, disjoint pieces and pipelined uploads. BitMate outperforms vanilla BitTorrent by 70% in download performance.

Figure 8 shows a comparison of upload contribution of BitMate with vanilla BitTorret. BitMate contributes 1000% more compared to vanilla BitTorrent.

4 COMPARISON WITH STRATEGIC CLIENTS

The optimizations implemented by BitMate result in a scrupulously honest client that contributes more than vanilla BitTorrent while improving its download bandwidth.

However, while we expected BitMate to perform better than vanilla BitTorrent, we were surprised to note that BitMate significantly outperforms strategic clients as well. In our comparison of BitMate with BitThief[3] and BitTyrant[2], we found that BitMate outperforms BitThief by a factor of 5 and BitTyrant by a factor of 3 in our target bandwidth of 5 kilobytes/sec.

We also run the experiments comparing BitMate with vanilla BitTorrent, BitThief and BitTyrant when the instrumented client is not in the lowest bandwidth group. In these experiments, we set the bandwidth of the instrumented clients at 20 kilobytes/sec (group 2 in our experiments). Even in this condition, BitMate outperforms BitThief by a factor of 3 and BitTyrant by as much as 60% on average (shown in figure 9).

The reason for these results is simple. Strategic clients are designed to minimize upload contribution. Since a low-bandwidth client is already struggling to find a place in the (reciprocal) unchoke list of other peers, further minimizing the upload contribution of a client makes the situation worse. In fact, on average, BitTyrant and BitThief perform even worse than vanilla BitTorrent for both 5 kilobytes/sec and 20 kilobytes/sec. For 5 kilobytes/sec, BitTyrant ends up

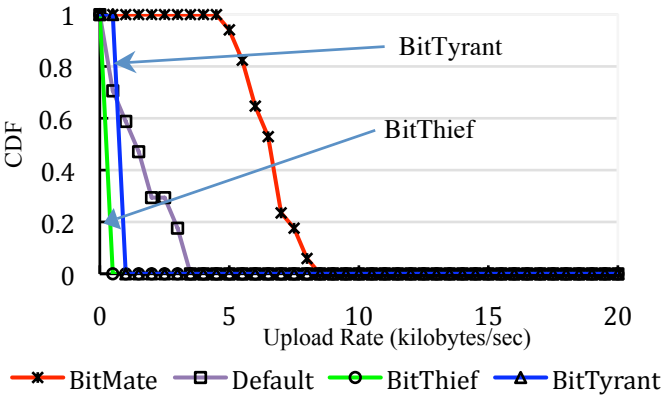

Figure 10: Upload contribution of BitMate compared with strategic clients.

contributing more than vanilla BitTorrent in its attempt to find the "sweet spot" for reciprocation, but still performs worse than both BitMate and vanilla bittorrent in download performance (upload contributions shown in figure 10).

5 RELATED WORK

A large body of work has analyzed, modeled and evaluated the performance, fairness and robustness of BitTorrent since Cohen's [5] seminal paper. Our work differs from previous work in one fundamental way: while the majority of previous work argues that BitTorrent is susceptible to free-riding by low-bandwidth nodes at the expense of high-bandwidth clients [2], we present a detailed evaluation to show that low-bandwidth

peers have no incentive to cheat. Instead a low-bandwidth client performs better if it uploads more. Our analysis also shows that BitTorrent's greedy peer selection policy, which favors higher bandwidth peers in a swarm, is actually unfair to low-bandwidth peers.

Recent models of BitTorrent, such as BitTyrant [2] argue that high-bandwidth nodes suffer due to their long convergence time to find matching peers, while low-bandwidth nodes utilize their bandwidth more efficiently [2]. In contrast, BitMate drives its performance by scrupulously implementing the fairness philosophy of BitTorrent.

Qiu and Srikant [11] presented a specific study of BitTorrent's rate-based TFT strategy, concluding that even in the presence of clients that upload at a reduced rate, the system converges to a Nash equilibrium where all peers upload at their capacity. Our experimental analysis, on the other hand, emphasizes that BitTorrent's incentive mechanism does not lead to an equilibrium, but instead behaves like a winner-takes-all auctions [7] in which low-bandwidth clients seldom win a bid.

A detailed simulation study of BitTorrent is presented by Bharambe et al. [10]. Though focused on high bandwidth clients, they propose two strategies to improve the fairness of BitTorrent by (i) matching peers with similar bandwidth, and (ii) enforcing tit-for-tat at the block level. BitMate implements a bandwidth-matching strategy and reports improvements in fairness as well as performance for low-bandwidth nodes.

The practicality of our approach makes it different from previous studies that propose schemes that are either not compatible with BitTorrent [7][10] or lack general appeal [2][3] (such as strategic clients that cheat to improve the performance of a peer at the expense of everyone else). Our approach, therefore, is fundamentally different from approaches that demonstrate performance improvement via Sybil attacks [12], uploading garbage data and expanding swarm size [3].

6 CONCLUSION

This paper makes three key contributions:

1. In contrast to a large body of recent work that argues that low-bandwidth clients benefit by strategic behavior, we demonstrate that there is no incentive for low-bandwidth clients to cheat since they actually perform better by enhancing their upload contribution.
2. Our analysis shows that low-bandwidth clients fail to *fairly* utilize their download bandwidth even when there are other peers in the swarm that can offer them good download performance.
3. Based on our analysis, we present a BitTorrent client called BitMate that is designed to enhance the performance of hosts with low-bandwidth connections by maximizing its upload contribution.

Overall, a low-bandwidth BitMate client prefers stable, bandwidth-matched peers over the greedy strategy of vanilla BitTorrent. Instead of wasting optimistic unchokes on high bandwidth peers, a BitMate client optimistically unchokes those peers that have a similar low-bandwidth. As a result, a BitMate client invests its scarce upload bandwidth on peers that are most likely to reciprocate. At the same time, BitMate leaves the tit-for-tat reciprocal unchoke policy untouched to uphold the fairness of BitTorrent. This leads to both increased performance and fairness since low-bandwidth clients can quickly form mutually beneficial peer-to-peer connections.

BitMate outperforms vanilla BitTorrent by as much as 70% in download performance, while at the same time improving upload contribution by an order of magnitude. BitMate also outperforms strategic clients like BitTyrant in low-bandwidth conditions by as much as 60% in download performance.

7 ACKNOWLEDGEMENTS

We would like to thank Omar Salman, Amina Khalid, Muneeb Waseem and Ghulam Murtaza for the initial experimental work that led to the design of BitMate.

8 REFERENCES

[1] Umar Saif, Ahsan Latif, Shakeel Farooq Butt, and Nabeel Farooq Butt, "Poor man's broadband: peer-to-peer dialup networking " In *ACM SIGCOMM CCR*, vol. 37, no. 5, October 2007

[2] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, Arun Venkataramani, "Do incentives build robustness in BitTorrent?" 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 2007)

[3] Thomas Locher, Patrick Moor, Stefan Schmid, Roger Wattenhofer, "Free Riding in BitTorrent is Cheap"*5th Workshop on Hot Topics in Networks (HotNets-V),* November 2006

[4] B Chun, D Culler, T Roscoe, A Bavier, "Planetlab: an overlay testbed for broad-coverage services", in ACM SIGCOMM 2003

[5] B Cohen, "Incentives build robustness in BitTorrent", Workshop on Economics of Peer-to-Peer systems, 2003

[6] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. 2007. "Clustering and sharing incentives in BitTorrent systems" In *Proceedings of the 2007 ACM SIGMETRICS*

[7] Dave Levin, Katrina LaCurts, Neil Spring, Bobby Bhattacharjee , "BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives", ACM SIGCOMM 2008

[8] Ningning Hu, Li Erran Li, Zhuoqing Morley Mao, Peter Steenkiste, Jia Wang, "Locating Internet Bottlenecks: Algorithms, Measurements, and Implications" , ACM SIGCOMM 2004

[9] BEP 30: Merkle hash torrent extension

[10] Bharambe et al. "Analyzing and Improving a BitTorrent Networks Performance Mechanisms", *25th IEEE International Conference on Computer Communications (INFOCOM 2006)*

[11] Dongyu Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks", ACM *SIGCOMM CCR*. 34, 4 (August 2004)

[12] Liogkas et al., "Exploiting BitTorrent For Fun (But Not Profit", *5th International Workshop on Peer-to-Peer Systems (IPTPS 2006)*