Improving Traffic Locality in BitTorrent via Biased Neighbor Selection

Ruchir Bindal
Pei Cao
William Chan
Department of Computer Science
Stanford University
rbindal@cs.stanford.edu

Jan Medved
George Suwala
Tony Bates
Amy Zhang
Cisco Systems, Inc.

Abstract

Peer-to-peer (P2P) applications such as BitTorrent ignore traffic costs at ISPs and generate a large amount of cross-ISP traffic. As a result, ISPs often throttle BitTorrent traffic to control the cost. In this paper, we examine a new approach to enhance BitTorrent traffic locality, biased neighbor selection, in which a peer chooses the majority, but not all, of its neighbors from peers within the same ISP. Using simulations, we show that biased neighbor selection maintains the nearly optimal performance of BitTorrent in a variety of environments, and fundamentally reduces the cross-ISP traffic by eliminating the traffic's linear growth with the number of peers. Key to its performance is the rarest first piece replication algorithm used by BitTorrent clients. Compared with existing locality-enhancing approaches such as bandwidth limiting, gateway peers, and caching, biased neighbor selection requires no dedicated servers and scales to a large number of BitTorrent networks.

1 Introduction

P2P content distribution applications such as BitTorrent [5] have fundamental advantages over the traditional client-server model (i.e. web sites) and the fixed-infrastructure content distribution networks (i.e. Akamai) as the supply of bandwidth grows linearly with the demand. They utilize the bi-sectional bandwidth among nodes while requiring little additional infrastructure. BitTorrent is therefore wildly popular and a major constituent of traffic on the Internet [7, 2].

An ISP typically pays a tier-1 "core" ISP for connectivity to the broad Internet, and traffic between the ISP and the outside world is costly for the ISP [18]. However, since the current implementations of BitTorrent ignore the underlying Internet topology or ISP link costs by setting up data transfers among randomly chosen sets of peers distributed over the Internet, it proves to be costly for ISP's.

ISPs often control BitTorrent traffic by "throttling", or bandwidth limiting. Since BitTorrent traffic typically runs over a fixed range of ports (6881 to 6889) [6] and is easily decoded, traffic shaping devices such as [20, 13, 19, 23] are deployed to limit the amount of bandwidth consumed by the BitTorrent protocol. However, this mainly slows down the content transfer and worsens the user download experience, not addressing the fundamental concern of the ISP, which is to improve the locality (i.e. reduce the cross-ISP traffic) of those transfers.

Many analytical and simulation studies [16, 1, 25, 22] have shown that the existing BitTorrent algorithm is nearly optimal in terms of user experienced download time. However, all these studies assume that a peer's neigbors are selected randomly from the set all neighbors. However, is random neighbor selection just a sufficient condition for performance optimality; or is it also a necessary condition?

This paper answers the above question with extensive simulations. We rely on simulations since it is difficult to capture all the relevant mechanisms in BitTorrent clients in an analytical model. Our simulator calculates network delays caused by multiple flows sharing a network link and accommodates internal network bottlenecks. The simulator models a peer's concurrent uploads, calculation of download rates from its neighbors, the choking/unchoking algorithm to decide who to upload to, and the piece selection algorithm. Different from other simulation studies [1, 15], the simulator does not assume that the bottleneck link is the individual peer's upload link, but rather can model arbitrary network links to be the bottleneck.

Our main conclusion is that biased neighbor selection, in which a peer chooses the majority, but not all, of its neighbors from peers within the same ISP, can reduce cross-ISP traffic significantly while keeping the download performance nearly optimal.

- As long as the original seed (the host that starts sharing the file) has moderately high upload bandwidth (e.g. four times the prevailing upload bandwidth of the peers), biased neighbor selection results in no degradation in download times, regardless of whether it is deployed by all ISPs or by just one ISP.

- Under biased neighbor selection, the cross-ISP traffic due to downloading of a particular file stays relatively constant, and does not grow as the number of peers interested in the file increases. In fact, as the number of peers inside an ISP increases, the traffic is reduced slightly.

- The "rarest first replication" algorithm is key to the success of biased neighbor selection. It has a "de-clustering" effect, improving the chances that peers within the same ISP have blocks to exchange with one another. Random piece selection, shown in other studies to work just as well in regular BitTorrent [1, 16], does not work well with biased neighbor selection. Here, "rarest first" refers to the strategy of choosing a block to download from the blocks a neighbor has. Under this scheme, the block which is least replicated among other neighbors is chosen first.

- For a cable modem or DSL ISP, when there are external high bandwidth peers, biased neighbor selection needs to be combined with bandwidth limiting to keep cross-ISP traffic at a minimum level. However, unlike pure bandwidth limiting, the combined scheme does not increase download time.

- Since each BitTorrent node prefers to upload to neighbors which have been giving data to it at a good rate, (the "tit-for-tat" mechanism) pure bandwidth limiting does push a peer toward intra-ISP peers, and can reduce cross-ISP traffic. However, its effect is limited by the initial neighbor selection of the peer, and combining bandwidth throttling with biased neighbor selection is much more effective.

- Allowing only a single peer to connect to the external nodes results in a significant increase in download time. This means that ordinary peers cannot act as gateway peers. Instead, dedicated high-bandwidth nodes should be used, and the approach does not scale to multiple concurrent BitTorrent networks.

- Using caches inside ISP's for BitTorrent traffic requires them to have high upload bandwidth to avoid increasing the download times. Combining biased neighbor selection with caches can reduce both the peak and average bandwidth needed, and the addition of bandwidth limiting can cap the peak bandwidth.

2 BitTorrent and Related Work

2.1 BitTorrent Protocols and Algorithms

BitTorrent [5] is a P2P file-sharing application which efficiently distributes large files to a large user population by making use of the **upload** bandwidth of all nodes (called peers) **downloading** the file. In the following description, the terms node and peer are used interchangeably.

To distribute a file via BitTorrent, the provider first generates a separate file containing some meta-information about the shared file (called the *torrent* file) and runs a central server called the *tracker* which keeps track of all nodes downloading this file. This *torrent* contains the address of the *tracker* and also the size of each of the small file blocks that will be exchanged between the peers. The supplier then starts its BitTorrent client(the *seed*). The torrent file is then published on the Internet using HTTP and interested downloaders can download it to run their BitTorrent clients with the torrent file as the input. Since the tracker's address is already embedded in the torrent file, the clients can contact the tracker using it. So the network consists of the tracker, the original seed, and all nodes interested in downloading the file. Peers are not fully connected to each other in this network. Rather, the network is a random graph.

Each peer, p, upon first joining the network, contacts the tracker. The tracker then **randomly** selects C nodes (default C is 50), out of all the nodes in the network, and hands the list back to p. Peer p then initiates connections with those nodes. Later on, other peers joining the network may get p as one of nodes returned by the tracker and initiate connections with p. By default, each peer connects to a maximum of 35 other peers on its own. Moreover, since peers may leave the network at any time, if p's neighbor count drops below a certain threshold (default is 20), p contacts the tracker again to obtain a new list of nodes. Note that the this description is of the prevailing default client implementation.

Peers exchange bit vectors of the blocks in their possession with neighbors frequently, both at the beginning and whenever a peer obtains new content. Through this bit vector exchange, the peer p learns the up-to-date content at each neighbor. If a neighbor has blocks that p doesn't have, p sends an "interested" message to the neighbor. Now, when and if the neighbor sends blocks to p depends on the "choking/unchoking" algorithm in BitTorrent.

Every 10 seconds, a peer evaluates which of its "interested" neighbors it will send data to. It *unchokes* 4 connections based on a "tit-for-tat" criteria where peers which have been giving data to it at a high rate are preferred. The fifth peer is *optimistically unchoked* i.e. is picked randomly from the remaining set of peers. Note that all connections are "choked"(or blocked) by default. This limit of 5 concurrent uploads is the default setting but can be configured to a different value. *Optimistic unchoke* allows brand new peers to bootstrap and also finds other peers which may have a better upload rate and is done every 30 seconds.

Clearly, the "tit-for-tat" mechanism biases the traffic towards higher bandwidth routes. However, this bias is limited by the initial neighbor selection that happens in the "forming the random graph" step.

On getting unchoked by a neighbor in which it is interested, a peer p chooses a block to download using the "local rarest first replication" algorithm. That is, among the blocks provided by a neighbor, the block that is least replicated among all neighbors of p is chosen. Note that a node only has visibility into the contents of its neighbors. Studies have shown that the rarest first selection algorithm is not necessary for the optimal performance of BitTorrent. However, as we will show later in the paper, it is in fact quite important if the graph is not random.

2.2 Existing Studies on BitTorrent

At its heart, BitTorrent attempts to solve the "broadcasting problem", i.e. disseminating M messages in a population of N nodes in the shortest time. Assuming a central scheduler and a completely-connected graph, studies [17, 25, 10] show that the lower bound on download time is $M + log_2(N)$ units, where a unit is the time it takes for two nodes to exchange a message.

BitTorrent does not support these assumptions but still appears to work exceedingly well. Simulation studies indicate that the random algorithms used by BitTorrent lead to a nearly optimal performance by utilizing upload links almost fully in a setting of cable modem and DSL nodes [1, 10].

A number of analytical studies support the above observation [16]. This analysis proves that, as long as the neighbors are chosen either randomly among all peers, or randomly among peers with the same number of blocks, the performance of BitTorrent is asymptotically optimal. The optimal performance does not depend on nodes staying around after completing their collections or using the least-replicated-first replication strategy.

All existing simulation and analytical studies, however, assume that peers choose neighbors randomly among all nodes in the network. Unfortunately, such neighbor selection policy is also the root cause of BitTorrent's high cross-ISP traffic. Thus, the goal of this study is to find neighbor selection policies that improve intra-ISP traffic locality while preserving the near-optimal performance of BitTorrent. The proposed solution, biased neighbor selection, achieves this goal.

In addition to bandwidth limiting, two other obvious methods for reducing cross-ISP traffic are caches [2], and "gateway peers" (a gateway peer is the only node inside an ISP that can connect to external peers) [15]. A recent trace-driven study examined the cross-ISP traffic of the two approaches and found that they are comparable [15]. However, the study did not look into peer download performance. We found that in order for these solutions not to increase download latency, the devices involved (caches or gateway peers) need to have much higher bandwidth than individual peers. In contrast, changing the neighbor selection policy requires no extra infrastructure, and can be combined with these methods to improve them further.

Measurement studies of BitTorrent traffic on the Internet [8, 14, 12] show that a BitTorrent network typically goes through three stages in its life: flash crowd, steady state and winding down [14], and the peer join rate decreases exponentially with time [12]. A flash crowd occurs when the content is first made available, and can last for several days [14]. Among the three stages, flash crowd generates the highest rate of traffic and is the most challenging for ISPs to handle. Thus, in this paper, the focus is on the flash crowd scenario, though the results are also validated with a Poisson arrival pattern.

Some other schemes using end systems for content distribution either build explicit overlay trees [9, 4, 3, 21] or meshes [3], and are quite different from BitTorrent. Slurpie [24] adopts the same randomized approaches as BitTorrent. We focus on BitTorrent due to its popularity.

Schemes using network coding to improve BitTorrent [11] solve the "last missing block" problem and significantly improve content availability in the network. In a sense, network coding is a replacement of the rarest first replication policy and is complementary to biased neighbor selection. We expect it can further reduce the cross-ISP traffic of biased neighbor selection, though a detailed study of the combination remains our future work.

3 Biased Neighbor Selection: Concept and Implementation

In biased neighbor selection, a peer chooses most of its neighbors from the same ISP as itself, and only a few from other ISPs. Specifically, a parameter k is associated with the scheme, where for each peer, all but k neighbors are from the same ISP, and only k neighbors are chosen from outside the ISP. If there are fewer than $35 - k$ internal peers (a peer by default keeps 35 neighbors), more external neighbors are retained. However, in this case, the peer contacts the tracker again at periodic intervals to find more internal neighbors.

Therefore, with biased neighbor selection, peers within the same ISP form a highly connected cluster. Each peer, however, keeps a few connections to the outside world, making the contents in the outside world visible to it. The difference between the normal randomized neighbor selec-

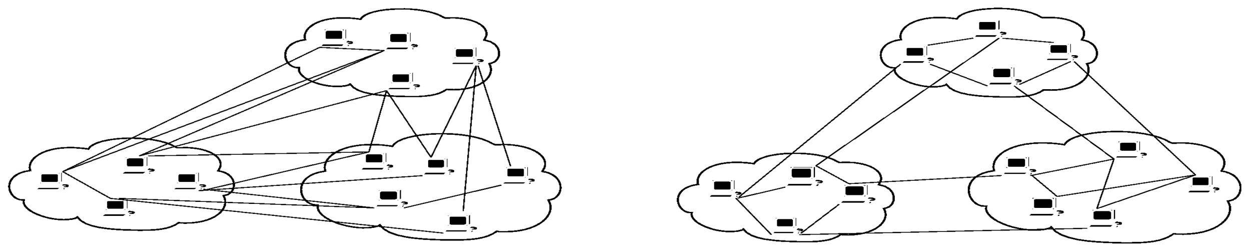

Figure 1. Uniform random neighbor selection vs. biased neighbor selection. The graph on the left is a result of uniform random neighbor selection, while the graph on the right is a result of biased neighbor selection.

tion and the biased neighbor selection is illustrated in Figure 1.

There are two ways to implement biased neighbor selection:

- *Modifying trackers and clients*: Biased neighbor selection can be implemented easily by changing the tracker and the client. The tracker selects $35 - k$ internal peers and k external peers to hand back to the client. If there are less than $35 - k$ internal peers, the tracker also notifies the client to contact it again after a certain duration. The challenge here lies in informing the tracker of the ISP locality. A number of options exist to solve the problem. The tracker can use Internet topology maps or IP to Autonomous System (AS) mappings to identify ISP boundaries. ISPs wishing to preserve traffic locality can also publish their IP address ranges to trackers. Finally, since BitTorrent tracker-client communication protocols run over HTTP [6], an ISP's HTTP proxy can append a new header "X-Topology-Locality" that contains a locality tag. All peers with the same locality tag can be assumed to be from the same ISP.
- *P2P traffic shaping devices*: In recent years, the need of ISPs to control P2P traffic has given rise to a new category of devices that we call P2P shaping devices. Situated along side the edge routers of the ISPs, these devices use deep packet inspection to identify P2P traffic and manipulate them. Representative vendors include CacheLogic [2], Sandvine [23] and Cisco's P-Cube appliances [19]. For BitTorrent traffic which runs on HTTP [6], many HTTP proxy appliances perform the role too. A P2P shaping device can implement biased neighbor selection without much obstacle. For each content file, it needs to keep track of peers inside the ISP downloading it. When a peer joins the network for a file, the device intercepts and modifies the responses from the tracker to the peer, substituting outside peers with internal peers. When it is necessary to change a peer's neighbors (for example, when more internal peers join the network), the device inserts TCP RESET on the connections between the peer and its external neighbors, forcing the peer to contact the tracker to obtain new neighbors. The device then manipulates the tracker's response to add internal neighbors.

Indeed, it is due to the prospect that biased neighbor selection can be implemented by P2P shaping devices without modifications to clients or trackers, that it is a practical method for ISPs.

4 Evaluation Methodology and Criteria

To understand the interaction of neighbor selection with other BitTorrent mechanisms, a discrete-event simulator was built to simulate the download of a large file over a simplified network topology.

4.1 Representative Network Topology

The basic network consists of 14 ISPs each having 50 peers joining a BitTorrent network. All peers inside the ISP are modeled after cable modem and DSL nodes, and have asymmetric upload/download bandwidths. The upload bandwidth of these peers is 100Kbps, and the download bandwidth is 1Mbps. These peers are referred to as "cable modem nodes" in this paper.

All ISPs are assumed to be completely connected. There are two scenarios to be considered, when ISPs do not impose bandwidth caps on BitTorrent traffic, and when ISPs do. Both are evaluated in this paper. In the case of bandwidth caps, the bandwidth cap is modeled as an ISP having a single limited bandwidth link to all other ISPs.

In addition to cable modem ISPs, the simulator considers the presence of high-bandwidth nodes that have symmetric links. We call these nodes "university nodes" for obvious reasons. The topology assumes that all university nodes have point-to-point links with each ISP and also with each other. The experiments vary both the number and the bandwidth of these nodes to examine their impact on the BitTorrent network.

Most of the simulations are conducted with all peers joining the network at once, i.e. the flash crowd scenario. The focus is on a flash crowd since it is the most challenging for ISPs to handle. Furthermore, it is assumed that all peers leave the network as soon as they finish download, but the original seed always stays online. Studies have shown that the majority of peers leave soon after they finish downloading [12]. The exception is the original seed. Since the content provider is interested in distributing the content using BitTorrent, it is reasonable to expect that the original seed stays around to see the last of the peers finish downloading.

4.2 Event-Driven Simulation

We built a discrete event simulator to calculate the download time of BitTorrent peers under various algorithms. The discrete event simulator models the following events in the BitTorrent network:

- Join: peers joining the network;
- Leave: peers leaving the network;
- Block-Transfer: a block is sent from one peer to another;
- Peer-Report: peer's periodic contact with the tracker;

Unlike simulators used in other studies, our simulator does not assume that the bottleneck is solely the upload links of the nodes, but rather accommodates bottlenecks in other network links. As a result, it needs to calculate the network delay of each block transfer based on the number of connections sharing bottleneck links and should also be capable of handling multiple bottlenecks.

The simulator calculates the network transfer delay in the following fashion. Every 100ms, the simulator recalibrates the transfer rate of each connection between peers, based on an idealized assumption of equal share and maximum capacity utilization. Each link has an associated upload and download bandwidth. For each link, the number of flows passing through it is recorded. Then, the simulator starts with the link that is most congested, i.e. with the lowest per-connection bandwidth, and sets the transfer rate for connections going over that link, assuming equal share. For each connection whose rate is set, the simulation goes through all links used by the connection and subtracts the connection's rate from those links. The leftover bandwidth of a link is assumed to be equally divided among the remaining connections. The simulator then finds the next most congested link, and repeats the above calculation.

This approach assumes idealized performance of TCP, and does not model the dynamics and idiosyncrasies of TCP implementations. Rather, we rely on the long-term fairness of TCP, documented by numerous analytical and measurement studies. The simulator also does not model the propagation delay of control messages, since they are negligible when compared to data transfer latencies.

The code simulating a peer is "fork-lifted" out of the original BitTorrent implementation. Data transfer events are translated into bytes received at the peer. Logic for calculating past download rates from the neighbors remains unchanged. The implementation of choking/unchoking algorithm is preserved. Each peer holds the bit-vector content of all its neighbors, and implements the "rarest first" replication algorithm.

4.3 Evaluation Criteria

There are two main evaluation criteria: download time and ISP traffic redundancy. Measurement of download time includes the cumulative distribution function (CDF), the 50^{th} percentile value and the 95^{th} percentile value. The size of the shared file is 64 MB in all experiment results presented in this paper. The file is divided into 2000 equal-sized blocks. Though a lot of files shared on real BitTorrent networks are much larger, a smaller file has been used in our experiments for the sake of speeding up the simulations. However, it was confirmed that running our simulator with a file 4 times as large does result in download time increased by a factor close to 4. So all results can be scaled for larger file sizes. The term "ISP traffic redundancy" means the average number of times each block of the content file travels into the ISP, until all peers inside the ISP finish their downloads. The lower the redundancy, the lower the cross-ISP traffic. The lowest redundancy is 1. The highest redundancy is N, where N is the number of peers inside the ISP.

It should be noted that for a file of size 64 MB, the minimum time required to fully upload it through a 100Kbps upload link is 5,120 seconds. This should also be the download time ideally experienced by all peers in a homogeneous network where every node has an upload bandwidth of 100Kbps. As is shown later, BitTorrent performs nearly optimally by bringing the download time very close to this number.

For each experiment, the simulation is run multiple times, with a different seed for the random number generator in each run. The variance of the results from multiple runs is very low, specifically $< 5\%$.

5 Biased Neighbor Selection: Benefits and Performance

The experiments examine two network settings: a homogeneous network consisting of 700 cable modem nodes spread among 14 ISPs, and a heterogeneous network consisting of a number of university nodes and 700 cable mo-

Table 1. Normalized download time and traffic redundancy under bandwidth throttling, in a homogeneous network. Download time for 1.0 is 5,312 seconds.

ISP bottleneck	50^{th} percentile	95^{th} percentile	Traffic redundancy
no bottleneck	1.0	1.35	46.9
2.5Mbps	1.43	1.59	31.76
1.5Mbps	2.01	2.05	24.88
500Kbps	3.33	3.53	21.65

dem nodes. In both cases, the original seed (the one provided by the content provider) is a separate node whose bandwidth varies. In the following discussion, we refer to the BitTorrent network using uniform random neighbor selection as "regular BitTorrent", and the BitTorrent network using biased neighbor selection as "biased BitTorrent".

5.1 Effects of Bandwidth Throttling

Bandwidth throttling by ISPs reduces the traffic redundancy moderately but causes a significant increase in download time, particularly when the seed has a high bandwidth or when there are external high bandwidth nodes. The performance with bandwidth throttling in homogeneous networks is shown in Table 1 and Figure 2. The seed is a separate node with a 400Kbps uplink bandwidth. For clarity, the download time results are presented as ratios between the download time and a base download time, which is the 50^{th} percentile download time for the no bottleneck case at 5,321 seconds.

In uniform random neighbor selection, the number of external neighbors that a peer has on average is $35*(1-N/G)$, where N is the number of peers in the ISP, and G is the total number of peers in the BitTorrent network. In homogeneous networks where there is no internal network bottleneck, all the neighbors have the same upload rate to the peer and hence they are indistinguishable from each other which leads leads to an expected redundancy of $N*(35*(1-N/G))/35 = N*(1-N/G)$. Our simulations match this analysis with 50 nodes per ISP and a total of 700 nodes.

Note that tit-for-tat causes the increase in download time to be less than linear to the reduction in bottleneck bandwidth. However, while a bottleneck of 1.5Mbps leads to halving the redundancy, setting the bottleneck to 500Kbps only reduces redundancy slightly further. It appears that beyond a certain level, bandwidth throttling cannot reduce traffic redundancy anymore because it is constrained by the initial neighbor selection.

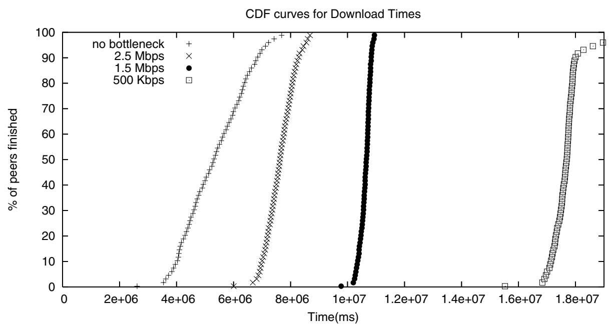

Figure 2. CDF of download times under bandwidth throttling, in homogeneous networks.

Table 2. Download time and traffic redundancy of regular vs. biased neighbor selection in a homogeneous network.

Neighbor Selection	50^{th} percentile	95^{th} percentile	Traffic redundancy
Regular BitTorrent	5,312	7,152	46.9
Biased $k = 1$	5,168	6,206	3.04
Biased $k = 5$	5,172	6,281	9.74
Biased $k = 17$	5,220	5,872	21.38

5.2 Biased Neighbor Selection in Homogeneous Networks

The above observation motivates our technique: biased neighbor selection. Namely, for BitTorrent traffic to have ISP locality, the neighbors need to be chosen well.

Table 2 and Figure 3 show the performance for biased neighbor selection, with the number of external neighbors k varying from 1 to 17 (half of all neighbors).

The results show clearly that biased neighbor selection reduces the variation in download times among the peers, has median download times similar to regular BitTorrent, and reduces ISP traffic redundancy significantly. The reduction in download time variances has to do with the fact that pieces are replicated more evenly in biased BitTorrent, since the estimate of replication ratios of pieces is improved due to the clustering properties of the graph. The variation in k has little impact on download time, and lower k results in lower redundancy. Thus, $k = 1$ should be used.

We also increased the number of peers inside each ISP, first to 75 and then to 100. Surprisingly, it was found that the redundancy *decreases* as the number of peers inside an ISP increases. The redundancy is 2.64 for 75 peers per ISP, and to 1.83 for 100 peers per ISP. We believe that as the number of peers inside an ISP increases, rare pieces brought from outside get more time to replicate inside the ISP as

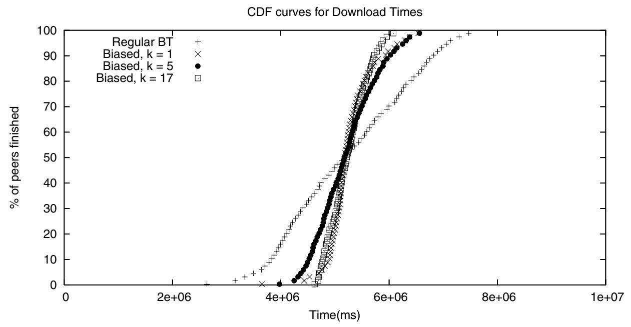

Figure 3. CDF of download times under biased BitTorrent in homogeneous networks.

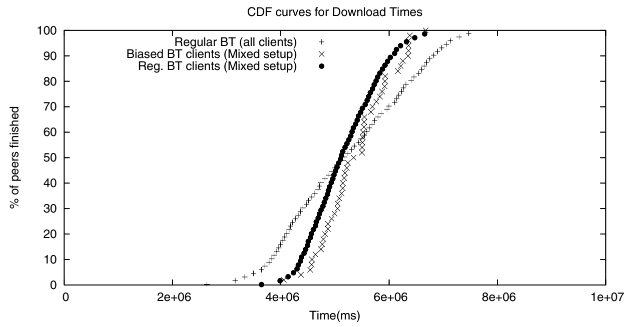

Figure 4. CDF of download times when only one ISP uses biased neighbor selection with $k = 1$.

compared to when all internal peers decide that a particular piece is rare and get it inside simultaneously.

The seed bandwidth was also increased from 400Kbps to 1Mbps. The traffic redundancy of biased BitTorrent stays virtually unchanged. Thus, in homogeneous networks, the traffic redundancy is mainly determined by the number of peers inside an ISP, and appears to stay under 4 in all cases.

Figure 4 shows the CDF of download times when only one ISP uses biased neighbor selection with $k = 1$. There is little change in median download time and the spread between download times is reduced for all peers. The ISP traffic redundancy is increased to 25.2 in this case, since the ISP using biased neighbor selection allows other ISPs to connect to nodes inside the ISP, and the average number of external neighbors for the ISP using biased neighbor selection is high. If the ISP allows only k connections per node from other ISPs, then the traffic redundancy can be reduced.

Overall, biased BitTorrent is a practical and effective solution at reducing cross-ISP traffic caused by BitTorrent. Any ISP using it realizes an immediate benefit.

The performance of biased neighbor selection, however, depends on the rarest first replication algorithm. Table 3 shows that the "de-clustering" effect of this algorithm which makes peers more interested in blocks that are rare inside the ISP.

Table 3. Effect of piece selection algorithms on biased BitTorrent. The download time of 1.0 means 5,168 seconds.

Piece Selection	50^{th} percentile	95^{th} percentile	Traffic redundancy
Random	1.84	2.51	14.4
Rarest first	1.0	1.20	3.04

Similar to other studies [1], we found that if the upload bandwidth of the original seed is 1x or 2x of cable modem peers' upload bandwidth, the average upload link utilization is around 50%. If the seed bandwidth is 400Kbps (4x cable modem peer's upload bandwidth) or higher, the average upload link utilization approaches 100% and biased neighbor selection in this case has no effect on download time. However, at a low seed bandwidth, the effect becomes unpredictable. We believe that since content providers can easily establish seeds with upload bandwidths 4 times that of an average node, low seed bandwidth cases are rare in practice.

Finally, in all our experiments, the original seed does not use biased neighbor selection. Since the goal of the original seed is to distribute contents to as many nodes as possible, it should not use biased neighbor selection.

5.3 Performance in Heterogeneous Networks

High-bandwidth "university nodes" (upload bandwidth of 400 KBps) were added to the above mentioned homogeneous network. Their number varies from 7 to 31, they leave as soon as finishing their download and do not use biased neighbor selection.

Table 4 and Fig. 5 show that biased BitTorrent seems to take advantage of the presence of these nodes sooner than regular BitTorrent. Furthermore, with 31 university nodes, biased BitTorrent outperforms regular BitTorrent slightly, mainly due to more uniform piece replications.

Understandably, the traffic redundancy of biased BitTorrent increases as the number of university nodes increases due to their high upload bandwidth. They are favored by the cable modem peers and also supply more blocks to them.

Table 5 shows that combining bandwidth throttle with biased BitTorrent in this case restores low redundancy and only increases download time slightly. At a 500Kbps bandwidth bottleneck, the traffic redundancy of biased BitTorrent is lowered to a value that is close to what is achieved

Table 4. Normalized download time and traffic redundancy of regular vs. biased neighbor selection as the number of high bandwidth peers increases. A download time of 1.0 is 5,312 seconds.

Extra University Nodes	Regular BitTorrent			Biased BitTorrent (k=1)		
	50th percentile	95th percentile	Traffic redundancy	50th percentile	95th percentile	Traffic redundancy
0	1.0	1.34	46.9	0.97	1.16	3.04
7	1.0	1.33	47.06	0.94	1.12	4.19
15	1.0	1.37	46.98	1.01	1.01	7.81
31	0.93	1.28	47.06	0.83	1.06	8.21

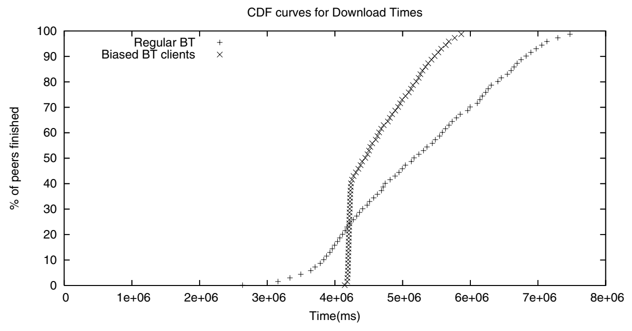

Figure 5. CDF of download times of regular BitTorrent vs. biased BitTorrent, with 31 high bandwidth nodes in the network.

Table 5. Combination of bandwidth throttling and biased BitTorrent in heterogeneous network. Download time of 1.0 is 4,446 seconds.

ISP bottleneck	50th percentile	95th percentile	Traffic redundancy
No bottleneck	1.0	1.27	8.21
2.5Mbps	1.10	1.33	6.74
1.5Mbps	1.09	1.32	7.37
500Kbps	1.12	1.34	4.40

when external high bandwidth nodes are not present, and the download time is increased only slightly ($< 12\%$). Yet, since most neighbors of a peer are within the same ISP and are not crossing the bottleneck link, the download time is not impacted much while the university nodes become less attractive.

Thus, throttling can be applied with biased neighbor selection without increasing download times significantly, regardless of whether the external peers have a high bandwidth. Our results also show that if an ISP deploys bandwidth throttling for BitTorrent traffic, then the peers inside the ISP can use biased neighbor selection to successfully avoid the bottleneck!

Table 6. Normalized download of the gateway peer approach, compared to regular BitTorrent, in homogeneous network.

Technique	50th percentile	95th percentile
Regular BitTorrent	1.0	1.34
100Kbps gateway peers	2.59	2.81
400Kbps gateway peers	1.0	1.14

6 Comparison with Other Locality-Enhancing Approaches

The above discussions show that biased neighbor selection performs much better than bandwidth throttling, and the two techniques can be combined for best results. This section examines two other techniques to reduce cross-ISP traffic: using a single peer, called "gateway peer", to connect to the external world [15], and using a cache to store blocks sent to the ISP [2].

An ISP can designate a single node inside it as the gateway peer. All peers inside the ISP can only connect to each other and to the gateway peer, but only the gateway peer can connect to the external world. However, as results from Table 6 show, gateway peers need to have an upload bandwidth that is at least 4x that of normal nodes in the in ISP to avoid increasing download times. Moreover, since the gateway peer has nothing to gain from the internal peers, because of the tit-for-tat mechanism it would rather exchange blocks with external peers, benefitting peers in other ISP's, causing them to finish faster (by as much as 20 % in one experiment). Finally, this approach is not scalable as the gateway requires 400KBps of upload bandwidth for each BitTorrent network that nodes from the ISP participate in.

Another approach to eliminate traffic redundancy is to use caches. Positioned at the ISP's gateway to the Internet, a cache stores blocks sent by external peers to internal peers, and when an internal peer wants to fetch a block from an external node, the cache intervenes transparently [2] and sends a locally-stored copy to the internal peer.

IEEE COMPUTER SOCIETY

Table 7. Peak and average upload bandwidth needs of caches.

Technique	Peak bandwidth	Average bandwidth
Caching (Regular BitTorrent)	3.61Mbps	1.73 Mbps
Caching (Biased BitTorrent)	1.32Mbps	153Kbps

Caches also need a high upload bandwidth to avoid increasing download times The estimated peak and average upload bandwidth needs of caches were obtained by summing up the bandwidth of flows crossing the ISP boundary that are "intervened" by the cache (see Table 7). Under regular BitTorrent, both the peak and average upload bandwidths of the cache are high. However, ISPs that deploy caches should instead use biased neighbor selection as the bandwidth needs of the cache are significantly reduced and furthermore, this can still be combined with bandwidth throttling.

7 Summary and Future Work

Our results show that BitTorrent networks can be clustered, as long as each node within each cluster keeps one or two connections to the outside, and rarest first piece replication is used. By clustering and biasing the peer connectivity graph, the linear growth of cross-ISP traffic in BitTorrent to the number of peers is eliminated, without an increase in download times. Biased neighbor selection can be combined with bandwidth limiting and caching to improve their performance further.

Our immediate next step is an implementation study. We are implementing the scheme in a modified BitTorrent client, and also in a HTTP proxy acting as a P2P shaping device. We are also in the process of obtaining approval to conduct BitTorrent experiments on the university network.

An intriguing potential of biased neighbor selection is that it offers a way for BitTorrent-like applications to bypass congestions on the Internet and optimize its performance. Toward this end, we plan to look into both simulations with more complex network scenarios, and analytical modeling and analysis to gain further insights.

References

[1] A. Bharambe, C. Herley, and V. N. Padmanabhan. Understanding and deconstructing bittorrent performance. In *Proceedings of SIGMETRICS*, 2005.

[2] CacheLogic. Cachelogic - advanced solutions for peer-to-peer networks. http://www.cachelogic.com.

[3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth content distribution in a cooperative environment. In *Proceedings of IPTPS'03*, 2003.

[4] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, pages 1–12, 2000.

[5] B. Cohen. Incentives build robustness in bittorrent, 2003.

[6] B. Cohen. Bittorrent documentation: Protocol, 2005. http://www.bittorrent.com/protocol.html.

[7] EContentMag.com. Chasing the user: The revenue streams of 2006, 2005. http://www.econtentmag.com/Articles/ArticleReader.aspx ?ArticleID=14532&ContextSubtypeID=8.

[8] P. G. Epema. The bittorrent p2p file-sharing system: Measurements and analysis.

[9] P. Francis. Your own internet distribution (yoid), 2002.

[10] P. Ganesan and M. Seshadri. On cooperative content distribution and the price of barter. In *Proceedings of ICDCS*, 2005.

[11] C. Gkantsidis and P. R. Rodriguez. Network coding for large scale content distribution. In *Proceedings of IEEE Infocom*, 2005.

[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measuremsnts, analysis and modeling of bittorrent-like systems. In *Proceedings of the Internet Measurement Conference*, 2005.

[13] C. S. Incorporated. Network based application recognition (nbar).

[14] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Hamra, and L. Garces-Erice. Dissecting bittorrent: Five months in a torrent's lifetime, 2004.

[15] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution. In *Proceedings of the Internet Measurement Conference 2005*, 2005.

[16] L. Massoulié and M. Vojnović. Coupon replication systems. *SIGMETRICS Perform. Eval. Rev.*, 33(1):2–13, 2005.

[17] J. Mundinger and R. Weber. Efficient file dissemination using peer-to-peer technology.

[18] W. B. Norton. The evolution of the u.s. internet peering system, 2003.

[19] P-Cube. P-cube: Ip service control. http://www.p-cube.net/indexold.shtml.

[20] Packeteer. Packetshaper. http://www.packeteer.com/prodsol/products/packetshaper.cfm.

[21] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking, 2002.

[22] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks, 2004.

[23] Sandvine. Sandvine: Intelligent broadband network management. www.sandvine.com.

[24] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol. In *Proceedings of IEEE Infocom*, 2004.

[25] X. Yang and G. de Veciana. Service capacity of peer to peer networks, 2004.