Zero-Day Reconciliation of BitTorrent Users with Their ISPs

Marco Slot[1], Paolo Costa[1,2], Guillaume Pierre[1], and Vivek Rai[1]

[1] VU University Amsterdam
[2] Microsoft Research Cambridge

Abstract. BitTorrent users and consumer ISPs are often pictured as having opposite interests, with end-users aggressively trying to improve their download times, while ISPs throttle this traffic to reduce their costs. However, inefficiencies in both download time and quantity of long-distance traffic originate in BitTorrent randomly selecting peers to interact with. We show that biasing the link selection allows one to reduce both median download times by up to 32% and long-distance traffic by up to 16%. This optimization can be deployed by modifying only the BitTorrent trackers. No external infrastructure nor specialized client-side software deployment is necessary, thereby facilitating the adoption of our technique.

1 Introduction

Peer-to-peer applications have significantly changed the landscape of Internet traffic management. While traditional client-server applications used to generate a majority of localized download traffic, peer-to-peer applications generate large amounts of global outgoing traffic. The impact is such that some fear the Internet will run into serious capacity problems within a few years [1]. In particular, BitTorrent, being reported as the foremost contributor of Internet traffic, has created a new antagonism between end users and their ISPs. While BitTorrent implementations deploy aggressive data transfer strategies to reduce file download times, consumer ISPs are forced to buy transit from global networks, driving up their operational costs. As a result some ISPs use throttling strategies to keep their costs under control [2], which in turn impacts the download times of their customers.

The reason why BitTorrent generates so much global traffic is that each peer of a given torrent selects other peers to exchange data with in a random fashion, without any consideration of network distance. Each peer then continuously updates its active connection set in a greedy fashion in favor of peers that can provide the best upload rates. The relative worse performance of long-distance peers may somewhat induce BitTorrent clients to exchange data with peers located nearby, but only among those in their local peer set.

This paper shows that careful selection of the initial peer sets given to each BitTorrent client can significantly reduce both the user-perceived download times and the generated amount of long-distance traffic. While this idea is not new, all

H. Sips, D. Epema, and H.-X. Lin (Eds.): Euro-Par 2009, LNCS 5704, pp. 561–573, 2009.

existing implementations rely on the global deployment of either network measurement infrastructures [3,4] or client-side extensions [5,6]. We argue that none of these approaches are practical, since they both require massive deployment of specialized software before beneficial effects become noticeable for the users and their ISPs. For example, while the authors of the Ono plug-in for Azureus can legitimately be proud of having deployed their code to over 400,000 clients, such a number represents only a small fraction of all BitTorrent users and therefore has little impact on global Internet traffic.

We propose to bias the connections maintained by BitTorrent clients towards nearby nodes using coordinate-based latency prediction. Importantly, our approach requires no global software adoption or deployment. The only required operation to optimize a whole torrent is to update its tracker with our software, and install a handful of globally distributed "landmark" servers. In our experiments we run a dozen of landmarks in PlanetLab, but in a real setting we expect that multiple BitTorrent trackers would organize themselves to become landmarks for each other. We refer to such reconciliation of both users and ISPs interests as *zero-day*, in reference to zero-day security attacks which can be launched at any time by mere exploitation of already deployed software.

Our approach relies on passive latency measurement. Peers are made to open TCP connections with each landmark by adding the addresses of landmarks to the first list of peers returned by the tracker. The measurements obtained from the TCP connection are used to compute the location of the peer in a centralized network coordinate system (GNP) [7]. The tracker can thus reply to any subsequent request from that peer with a list of carefully selected peers in place of the usual random choice. Applying this selection bias significantly reduces traffic cost for ISPs, while reducing download times for the tracker's end-users. We evaluate our approach through carefully designed simulations, which we validate against PlanetLab. We show that it allows to reduce the median download times by up to 32%, and the quantity of global Tier-1 traffic generated by up to 16%.

2 Related Work

Several approaches proposed to bias BitTorrent traffic towards nearby peers. However, they all rely on large external infrastructures or client-side modifications, which largely hampers their applicability.

Bindal *et al.* proposed bias based on peers' ISPs [5]. A peer selects k peers within the same ISP and the rest from other ISPs. Unfortunately this requires either modified BitTorrent clients or specialized infrastructure at the ISPs which makes deployment difficult. Moreover, the whole approach depends on a high number of peers sharing the same ISP, which rarely happens in practice. Finally, this approach does not distinguish between two ISPs in the same town, which are likely to have some peering relationships in place, and two ISPs in different continents, which, instead, must acquire transit from Tier-1 ISPs in order to communicate.

The iPlane project monitors routes from hundreds of locations to build a single, coherent view of the Internet [4]. iPlane can help build a biased BitTorrent tracker. Rather than using only network distance, it also uses loss rates to estimate the TCP throughput between peers. The fine-grained network measurements and models of iPlane may produce more accurate results than GNP, but they require a huge infrastructure. This infrastructure is currently deployed on PlanetLab for research purposes, but deploying a commercial or public alternative would be much more expensive than a cooperative GNP system. Finally, the iPlane topology maps are very large (several GBytes) and look-ups computationally expensive.

P4P allows ISPs to publish their network information and preference for peers to connect with [3]. This solution exploits a network oracle akin to iPlane but here the ISPs can provide fine-grain network information. P4P has support from a few large ISPs, but to be effective it would need to be supported by a vast majority of ISPs worldwide and in collaboration with BitTorrent trackers.

The Azureus BitTorrent client implements the Vivaldi network coordinate system [8]. Vivaldi strongly resembles GNP, but uses peer-to-peer interaction instead of fixed landmarks. Although this approach could allow Azureus clients to bias the choice of peers to unchoke, this technique is limited to Azureus clients only.

Some public databases (e.g., `http://www.ripe.net`) enable to map IP addresses to ISPs, thus possibly removing the need of tracking peer's location through landmarks. This information, is however often incomplete and coarse-grained. Similarly to [5], it does not allow to identify nearby peers from different ISPs.

Finally, Ono is a plug-in for the Azureus BitTorrent client that relies on Akamai [6]. Akamai owns servers in many locations, and redirects its clients to the closest replica through DNS. Each Ono peer resolves DNS names of popular websites hosted by Akamai. If two peers receive the same IP addresses, they are likely to be relatively co-located. Ono uses this information to select nearby peers. Ono's applicability is however unclear. Although it has been downloaded 400,000 times, this represents only a small fraction of all BitTorrent users. Hence, any gains made due to selection will only have local effects. In addition, the authors show that Ono-enabled clients experienced slightly *lower* median download rates than regular ones.

3 Background

3.1 BitTorrent

BitTorrent is a peer-to-peer protocol for distributing large files [9]. Each file offered for distribution uses a separate overlay, managed by one tracker responsible for maintaining the overlay membership. To join a BitTorrent network, a client registers at the corresponding tracker, with the identity of the torrent to join and its own contact address[1]. The tracker adds the new peer to the membership

[1] Although some recent versions of BitTorrent clients also support a DHT-based decentralized tracker, most available torrents normally rely on a centralized tracker [10].

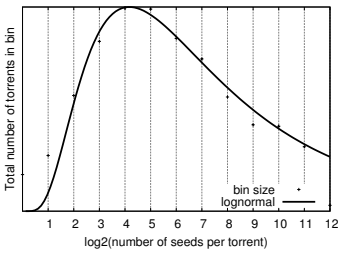


Fig. 1. Swarm size distribution (number of peers in swarms with $2^{x-1} < size \leq 2^x$)

list, and returns a random subset of this list to the client to build its initial *peer set*.

BitTorrent relies on incentives to encourage participants to contribute their upload bandwidth in the file distribution process. The incentive scheme is based on reciprocation such that a peer uploads content only to those from which it receives something in return. Each peer autonomously decides which peers to *unchoke*, that is which peers to send file content to. Similarly, each peer can decide to *choke* a peer that does not upload sufficient data and unchoke another peer in the hope to obtain a better throughput. BitTorrent thus uses only local greedy policies in selecting the destination of uploaded traffic so as to maximize its own download rate. Other possible optimization metrics such as the implied cost of long-distance traffic are not taken into consideration. The local peer set is refreshed only when peers disconnect due to client exit, connection failure or long period of inactivity. This makes the initial peer selection particularly critical to provide the peer with a good set of nodes to choose from.

Importantly, any form of a peer selection bias can be effective only when applied to large enough networks. If the swarm is smaller than the peer set size (usually 50), clients will have all peers in their peer set. We studied the distribution of network sizes that any peer may be part of, by screen-scraping the number of peers from 150,000 torrents on a popular BitTorrent tracker. Figure 1 shows that about half of the clients belong to a network of 82 peers or more, and are therefore likely to benefit from biased link selection. In addition, we can expect the networks they are in to generate the majority of traffic as many smaller networks are idle.

3.2 ISP Economics

ISPs use two types of relationship to carry traffic between machines located in different ISPs. First, two ISPs that exchange large amount of traffic may *peer* with each other, that is establish a direct connection between their two networks. The cost of a peering relationship is virtually constant regardless of the quantity of traffic exchanged. Other traffic is sent through a *Tier-1* ISP, whose business is to carry traffic between ISPs. Tier-1 ISPs charge their service on a per-volume

and sometimes per-distance basis. Consumer ISPs therefore have a financial incentive to reduce the volume of traffic that their users upload into Tier-1 ISPs.

BitTorrent, by uploading large quantities of traffic to randomly selected nodes creates a financial burden to consumer ISPs. Since peering can only be done between ISPs that are physically close, most of this traffic must be sent over Tier-1 networks. Some ISPs have reportedly tried to reduce their costs by throttling BitTorrent traffic. Although this may indeed reduce their peak traffic rate, it creates bad customer experience. A better solution in our opinion is to deploy mechanisms such that the majority of BitTorrent traffic is exchanged inside ISPs or through peering relationships instead of Tier-1 ISPs. As we show later, such measures have the favorable side effect of reducing user-perceived BitTorrent download times.

3.3 Peer-to-Peer Connection Throughput

Download time improvements derive from the fact that reducing the latency of paths has the side-effect of improving the connection throughput. We consider three main parameters that together contribute to defining the throughput in BitTorrent.

Access link capacity. Many users access the Internet through asymmetric cable or ADSL connections. This has important implications for BitTorrent where each downloaded packet is in principle reciprocated with one uploaded packet: download link capacities are rarely saturated by BitTorrent traffic. The only bandwidth bottleneck concerns the upload capacity. Even though this limitation remains constant regardless of the BitTorrent node selection strategy, upload capacities are not necessarily utilized to their full extent in real scenarios, and potentially provide room for download time improvement.

Internet throughput. Although the core Internet links can be considered as well provisioned compared to the end users access links, there exists a weak, inverse correlation between latency and bandwidth [11, 12]. Therefore, reducing the latency of paths used for BitTorrent traffic can likely increase the throughput of these connections. Similarly, shorter paths are likely to exhibit lower packet loss rates.

TCP throughput. Any single TCP connection has a maximum achievable throughput driven by inter-node latency, and regardless of the capacity of the underlying network. This bottleneck is due to the fact that TCP window sizes are limited to 65536 bytes. In the total absence of packet loss, the maximum throughput of a single connection is $\frac{W_{max}}{RTT}$, where W_{max} is the maximum window size and RTT is the round-trip time. Any packet loss further reduces this maximum throughput.

4 Design and Implementation

Our approach relies on instrumenting the BitTorrent tracker so that it can estimate inter-peer network latencies, and return biased selections of links to each

peer. Each time a peer P connects to the tracker, the tracker returns a set of other peers which exhibit low latency paths to P. The client remains completely unaware of this bias and behaves as usual. However, as we show in the next section, due to the low latency of these paths, these connections are more efficient than in the traditional BitTorrent and enable saving download time and reducing inter-AS hops.

To build this, one should address three questions: (i) the tracker must estimate inter-peer latencies, although issuing N^2 measurements in a torrent with N peers is unacceptable; (ii) latency measurements must be realized with no explicit support from the peers themselves to preserve our client-independent approach; (iii) the tracker must select links to return to each peer so as to favor low latencies without compromising other important properties of the BitTorrent swarm.

4.1 Latency Estimation

Directly measuring the pairwise latencies between peers of a given torrent would require $O(N^2)$ measurements, which is unacceptable for large torrents. A classical solution to this problem is the use of GNP "network coordinates" where each node is given a d-dimensional coordinate and inter-node latency is estimated as the Euclidean distance between coordinates [7,13]. The advantage of coordinate-based latency prediction is that it only requires latency measurements from each node to $d + 1$ landmarks, as opposed to N^2 for pairwise measurements.

In the initial phase, each landmark measures its round-trip-time to every other landmark. Landmark coordinates can then be computed such that the Euclidean distance between coordinates matches the measured latencies. This translates into a minimization problem over an error function ε. The procedure to determine the coordinates for a regular node is similar to the initial phase. The round-trip-time between the node and each landmark is measured and coordinates are computed by minimizing the sum of the error function. GNP can predict over 90% of latencies within 50% relative error. This is generally sufficient to find nearby hosts.

4.2 Passive Latency Measurements

Each time a new peer joins a torrent, we need to issue latency measurements between the peer and each of the landmarks to derive the new peer's coordinate. However, we cannot expect any explicit support from the peer itself since we do not want to rely on specialized software at the client side. Instead, we rely exclusively on the BitTorrent protocol itself and on passive latency measurements.

Whenever a new BitTorrent client registers at the tracker, the tracker returns a small set of randomly selected peers (like a normal tracker), plus the addresses of the landmarks. Since clients start with an empty peer set and do not distinguish between actual peers and the landmarks they will open connections to each. A landmark can then passively measure its latency to the peer by measuring delays between packets during the TCP three-way handshake [14]. Once a connection is established, the landmark gracefully closes the connection and reports the measured latency to the tracker. When the tracker has received

enough measurements from the landmarks, it can compute the peers' coordinate. The client, after several unsuccessful connections to landmarks, will contact the tracker again to obtain a fresh set of peers. The tracker can then predict the latency between the client and other peers using the Euclidean distance between the nodes' coordinates.

While a tracker may decide to run landmarks itself, we expect small groups of trackers to organize and provide each other with landmarks for their mutual benefit. The associated workload is very low. Each client must be measured once by each landmark, and the implied network overhead is very low. Trackers can also maintain a memory of past measurements, which further reduces the overhead due to latency measurements [13].

4.3 Peer Selection

When a BitTorrent client contacts the tracker for new peers, the tracker should not systematically return the n closest peers to this client. First, doing this creates a risk of partitioning the swarm into disjoint cliques. Second, it would reduce the interest for a client to ask for new peer sets, since these sets would remain largely identical from one request to another.

Our modified tracker returns a number of peers selected randomly from the 25% closest peers, plus a few more peers selected randomly in the whole swarm. The first measure increases the gain of repeatedly contacting the tracker, while the second keeps the swarm connected. We however note that in our experiments we found it extremely difficult to partition BitTorrent swarms, even when the tracker returns no long-distance link. We discuss this further in Section 5.5.

Another potential issue is that, when a new torrent is created, the tracker initially has only few nodes that can be returned to the clients. If the tracker would return a full peer set, this would create a clique of nodes all connected to each other. Any subsequent node that joins the torrent would find it difficult to connect to any pre-existing node. We prefer returning a smaller number of links to the first clients that show up, so that subsequent nodes can join more easily.

5 Evaluation

Evaluating a BitTorrent optimization in a realistic setting is very challenging due to the difficulty of involving hundreds of users across the world. We evaluate our system on PlanetLab, although the large bandwidths between PlanetLab nodes are not representative of most BitTorrent users. We therefore also developed a simulator that reproduces the network conditions experienced by regular BitTorrent clients, communicating over the Internet. The simulator models propagation delay, TCP throughput and upload capacity sharing. We feed it using data obtained from actual BitTorrent clients and use PlanetLab experiments to *validate* the simulator[2].

[2] All our implementations can be found at `http://marcoslot.net/latorrent.htm`

The simulator implements the standard BitTorrent algorithms [15,16], including choking, optimistic unchoking, strict piece priority, request pipelining and rarest first piece selection. Messages are delayed based on latency data and TCP throughput is estimated after the popular PFTK model [17] with a window size of 2^{16} bytes, taking into account fair sharing of the access link capacity.

5.1 Experimental Settings

We first present a set of experiments using real measurements taken from PlanetLab. We focus here on validating our simulator by comparing the results obtained in both approaches. We deployed the original BitTorrent client (version 5.2.0) on 141 PlanetLab nodes, and our landmark implementation on 7 nodes. We fed our simulator with the same configuration using measured latencies and packet loss rates between these nodes.

We tested our system under two scenarios with maximum sizes of 200 and 1,000 nodes to analyze the impact of biased selection in small and large-scale BitTorrent networks. To extract a representative set of peers to use in our simulations we used the data provided by iPlane [4], a service providing accurate loss, latency and path predictions for a large number of Internet hosts. iPlane periodically participates in BitTorrent swarms to also measure access link bandwidths. From the resulting set of BitTorrent peers and their pairwise latency, loss and path predictions we used uniformly drawn samples in the simulations.

To reproduce a realistic join rate, both in the PlanetLab- and iPlane-based experiments, we took segments out of the Izal tracker log [18]. We introduce new peers based on the offset of the starting times in the log and continue doing so until we reach the maximum number of peers. The peers download a 256MB file originating from a single seed, which remains available during the whole experiment. The seed has above-average upload capacity. Since our completion times will not match the tracker log we do not use the tracker departure times. Instead we let peers stay with mean departure time of 2 minutes after completing their download.

We measure the following metrics: (i) the *download time*, defined as the time required for a peer to download the whole file; (ii) the *latency* of network paths, weighted by the quantity of traffic actually exchanged via each path; and (iii) the fraction of traffic exchanged through a *Tier-1* ISP. We considered ASNs 174, 209, 701, 1239, 1299, 1668, 2914, 3356, 3549, 3561, 6453 and 7018 as Tier-1 ISPs.

Each experiment was repeated 5 times in each configuration.

5.2 Simulator Validation

To validate the accuracy of our simulator on a real wide-area network, we run our system in PlanetLab and compared the results against those obtained from the simulator, fed with the real values of latency, bandwidth and packet-loss rate as measured on PlanetLab. Table 1 shows that our simulator is indeed successful

Table 1. PlanetLab vs. Simulated performance

	Download time			Latency		
	PlanetLab	Simulated	Δ(%)	PlanetLab	Simulated	Δ(%)
Standard(median)	357 s	347 s	2.08%	57 ms	49 ms	14.0%
Biased (median)	286 s	288 s	0.7%	55 ms	45 ms	18.2%
Standard (90-th percentile)	1700 s	1704 s	0.2%	268 ms	254 ms	5.2%
Biased (90-th percentile)	1251 s	1387 s	9.2%	267 ms	240 ms	10.1%

(a) Download time and latency

	PlanetLab	Simulated	Δ(%)
Standard	41%	39%	4.8%
Biased	39%	38%	2.5%

(b) Tier-1 Traffic

at reproducing the behavior of a real network, providing a good approximation of the real performance for all three metrics introduced above[3].

This is a key result for two reasons. First, it allows us to concentrate on simulations to evaluate the effectiveness of our approach in settings typical of real BitTorrent swarms. Second, it shows that our biased tracker reduces the median download rate by 20% and the 90-th percentile by 26%. Conversely, the improvement is lower in terms of Tier-1 traffic. The reason is that most traffic in PlanetLab is routed through few different university networks, so the incidence of Tier-1 traffic is relatively limited. As we will observe later in this section, in realistic settings the Tier-1 traffic accounts for more than 80% of the whole traffic.

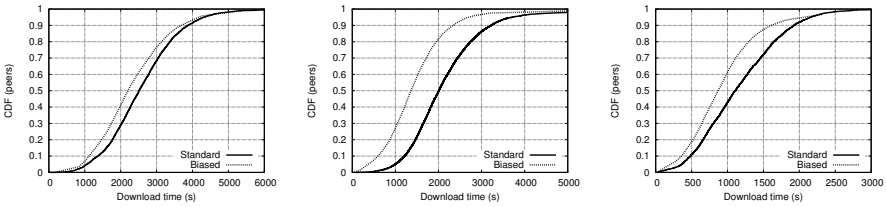
5.3 Simulation Results

We now turn our attention to the evaluation of our approach based on simulations, and using a more realistic set of nodes and network metrics taken from iPlane.

Tier-1 traffic. Our goal is to reduce both the cost of ISPs and the user download times. To analyze the cost of ISPs we determine how much traffic is routed through a Tier-1 ISP. Table 2 confirms that a large fraction of standard BitTorrent traffic is routed through Tier-1 ISPs, and therefore results in a direct financial cost for consumer ISPs. However, using the biased tracker the Tier-1 traffic is reduced by 11% for a network of 200 peers and by 16% for a network of 1000 peers.

Note that these results have been obtained using a uniformly random sample of nodes from the iPlane dataset. This means that nodes are uniformly spread around the globe, therefore making Tier-1 traffic unavoidable. However, many torrents are only of regional interest [19]. We expect that our approach would

[3] Note that even though the *relative* value of the median error for the latency is rather high (∼15%), its *absolute* value (<10ms) makes it practically negligible.



(a) 1 seed, 200 leechers (b) 1 seed, 1,000 leechers (c) 100 seeds, 100 leechers

Fig. 2. Number of peers with download time $< x$

Table 2. Percentage of traffic routed through Tier-1 ISPs

Network size	Standard	Biased	Gain(%)
200 nodes	81%	73%	11%
1,000 nodes	81%	68%	16%

perform better in such settings, since the tracker would have more short paths to choose from.

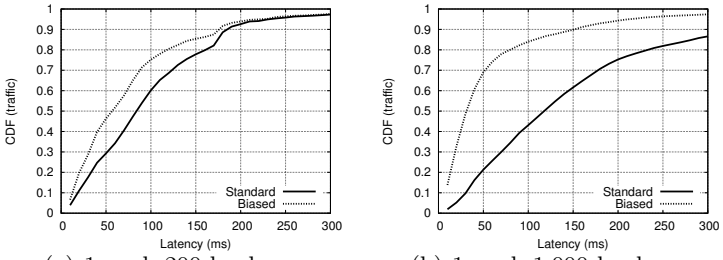
Download Time. Figure 2 reports the CDF of download times for a 200-node and a 1000-node network. The biased approach delivers much better download times, with improvement of the median download time of respectively 12% and 32%. Large networks perform better because the tracker then has more nodes to choose from.

These two graphs represent a worst-case scenario in the sense that the swarm has only a single seed. However, real torrents normally have several seeds available at any time. In the data from Figure 1 we observed on average 5 seeds for every 3 leechers. Figure 2(c) shows a 200-nodes scenario where the swarm contains 100 seeds and 100 leechers. The median gain in download time raises from 12% to 22%.

Traffic Latency. To estimate the distance over which traffic is sent, Figure 3 shows the CDF of latency experienced by traffic. Results show a large decrease of latency when using the biased tracker, with median improvements of respectively 33% and 75%. Again, the large-scale network exhibits a greater improvement.

5.4 Adaptive Sample Size

In previous experiments, the tracker always returned a full peer sample size (50 peers). However, as discussed in Section 4.3, this can degrade the performance because early nodes may establish links to distant nodes and later prevent closer nodes to contact them. We now evaluate our system when the size of the sample returned is $2 \times \sqrt{N}$, where N is the current swarm size.



(a) 1 seed, 200 leechers (b) 1 seed, 1,000 leechers

Fig. 3. Traffic exchanged over links with latency $< x$

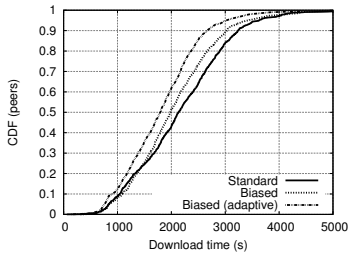


Fig. 4. Number of peers with download time $< x$ (1 seed, 200 leechers)

Figure 4 evaluates the adaptive strategy against the standard and biased ones in the 200-leechers scenario. To better isolate the phenomenon we focus only on the first 200 peers which completed the download. Clearly, performance benefits from the adaptive solution. While the biased tracker achieves a gain of 12% compared to the standard tracker, adapting the sample size to the current size of the network gains 18%. Even an unbiased tracker shows a small gain using this technique.

5.5 Bias Degree

As detailed in Section 4.3, our biased tracker returns a few random nodes (10% in our experiments) to prevent network partitions. Figure 5 shows the effect of varying the bias degree in the 200-leechers scenario, ranging from a fully random selection (bias=0) to a fully biased one (bias=1). Even with a full bias no network partitioning appears, and our system performs even better. The reason is that the default size of the peer set (50 nodes) is large enough to ensure strong connectivity of the entire network. On the other hand, for lower values of the peer set, we can foresee that partitions may occur. Investigating the impact of bias using different values of the peer sets is part of our immediate research agenda.

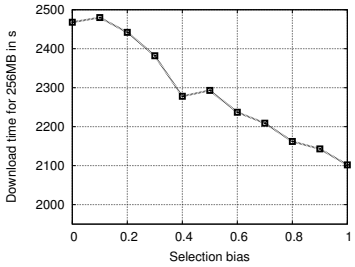


Fig. 5. Median download times for different bias degrees

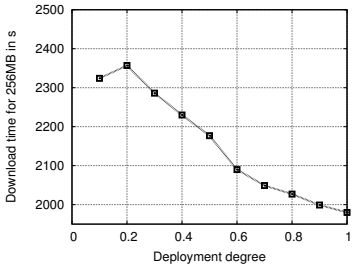


Fig. 6. Median download times for a client-based implementation against different deployment degrees

5.6 Deployment Degree

We claim that modifying only the tracker allows to optimize entire swarms at a time, while client-based approaches depend on the number of clients which implement the latency-based bias. Figure 6 shows the performance of an Ono-like approach [6] for different fractions of Ono-enabled clients in the swarm. It is difficult for us to reproduce the accuracy of Ono's latency estimations. Instead, we assume here that enabled clients have a *perfect* latency predictor among themselves to select nearby links. We observe that, despite the very favorable assumptions taken, in order to achieve comparable performance to our approach, more than 50% of clients must participate. In real settings, with inaccurate latency prediction, this number would reasonably be even higher. Given the vast number of BitTorrent implementations, this would be very hard to achieve in practice.

6 Conclusions

BitTorrent encourages peers to donate their unused networking resources to help distribute popular content. The tremendous success of this simple idea however sharply increased the amount of global traffic uploaded by end users. We showed in this paper that relatively simple optimizations can provide significant performance improvements, with 11-16% reduction of expensive Tier-1 traffic in challenging scenarios. We also find the median download times are reduced by 12-32%, providing strong incentive to deploy these optimizations.

We explicitly aimed at easy deployment of our approach. We therefore excluded any solution that would require code deployment at the client side, or the prior availability of massive global services. Instead, optimizing a whole torrent requires only a modified tracker and a handful of landmarks. We even expect that the need for explicit landmark deployment will disappear if multiple trackers agree to cooperate and behave as each other's landmarks.

We present this work and make our implementations freely available in the hope to attract the attention of major BitTorrent tracker operators. Should they adopt this technique, the seemingly opposite interests of BitTorrent users and their ISPs might be partially reconciled.

Acknowledgments

We wish to thank Harsha Madhyastha for providing access to the iPlane data [4] and Michal Szymaniak for his SYNACK/ACK and GNP implementations [13].

References

1. CNET: AT&T: Internet to hit full capacity by 2010, http://news.cnet.com/ATT-Internet-to-hit-full-capacity-by-2010/2100-1034_3-6237715.html
2. CNET: Comcast on the hot seat over BitTorren, http://news.cnet.com/Comcast-on-the-hot-seat-over-BitTorrent/2009-1025_3-6231975.html
3. Xie, H., Yang, Y.R., Krishnamurthy, A., Liu, Y.G., Silberschatz, A.: P4p: provider portal for applications. In: Proc. SIGCOMM (2008)
4. Madhyastha, H.V., Isdal, T., Piatek, M., Dixon, C., Anderson, T.E., Krishnamurthy, A., Venkataramani, A.: iPlane: An Information Plane for Distributed Services. In: Proc. OSDI (2006)
5. Bindal, R., Cao, P., Chan, W., Medved, J., Suwala, G., Bates, T., Zhang, A.: Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In: Proc. ICDCS (2006)
6. Choffnes, D.R., Bustamante, F.E.: Taming the Torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In: Proc. SIGCOMM (2008)
7. Ng, T.S.E., Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches. In: Proc. INFOCOM (2002)
8. Dabek, F., Cox, R., Kaashoek, M.F., Morris, R.: Vivaldi: a decentralized network coordinate system. In: Proc. SIGCOMM (2004)
9. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Proc. First Workshop on Economics of Peer-to-Peer Systems (2003)
10. Crosby, S.A., Wallach, D.S.: An Analysis of BitTorrent's Two Kademlia-Based DHTs. Technical Report TR-07-04, Rice University (2007)
11. Oppenheimer, D., Chun, B., Patterson, D., Snoeren, A.C., Vahdat, A.: Service placement in a shared wide-area platform. In: Proc. USENIX Technical Conf. (2006)
12. Lee, S.J., Sharma, P., Banerjee, S., Basu, S., Fonseca, R.: Measuring Bandwidth Between PlanetLab Nodes. In: Proc. 6th Intl. Workshop on Passive and Active Network Measurement (2005)
13. Szymaniak, M., Presotto, D., Pierre, G., van Steen, M.: Practical large-scale latency estimation. Elsevier Computer Networks 52(7) (2008)
14. Jiang, H., Dovrolis, C.: Passive estimation of TCP round-trip times. In: Proc. SIGCOMM (2002)
15. bittorrent.org: Bittorrent specification, http://www.bittorrent.org/beps/bep_0003.html
16. theory.org: Unofficial BitTorrent specification, http://wiki.theory.org/BitTorrentSpecification
17. Padhye, J., Firoiu, V., Towsley, D.F., Kurose, J.F.: Modeling TCP throughput: A simple model and its empirical validation. In: Proc. SIGCOMM (1998)
18. Izal, M., Urvoy-Keller, G., Biersack, E.W., Felber, P., Hamra, A.A., Garcés-Erice, L.: Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In: Proc. 5^{th} Intl. Workshop on Passive and Active Network Measurement (2004)
19. Iosup, A., Garbacki, P., Pouwelse, J.A., Epema, D.H.J.: Three lessons from one peer-level view. In: Proc. 11^{th} Conf. of the Advanced School for Computing and Imaging (2005)