TopBT: A Topology-Aware and Infrastructure-Independent BitTorrent Client

Shansi Ren[1], Enhua Tan[2], Tian Luo[2], Songqing Chen[3], Lei Guo[4], and Xiaodong Zhang[2]
[1]Microsoft Corporation
[2]The Ohio State University
[3]George Mason University
[4]Yahoo! Inc.

***Abstract*—BitTorrent (BT) has carried out a significant and continuously increasing portion of Internet traffic. While several designs have been recently proposed and implemented to improve the resource utilization by bridging the application layer (overlay) and the network layer (underlay), these designs are largely dependent on Internet infrastructures, such as ISPs and CDNs. In addition, they also demand large-scale deployments of their systems to work effectively. Consequently, they require multi-efforts far beyond individual users' ability to be widely used in the Internet.**

In this paper, aiming at building an infrastructure-independent user-level facility, we present our design, implementation, and evaluation of a topology-aware BT system, called TopBT, to significantly improve the overall Internet resource utilization without degrading user downloading performance. The unique feature of TopBT client lies in that a TopBT client actively discovers network proximities (to connected peers), and uses both proximities and transmission rates to maintain fast downloading while reducing the transmitting distance of the BT traffic and thus the Internet traffic. As a result, a TopBT client neither requires feeds from major Internet infrastructures, such as ISPs or CDNs, nor requires large-scale deployment of other TopBT clients on the Internet to work effectively. We have implemented TopBT based on widely used open-source BT client code base, and made the software publicly available. By deploying TopBT and other BitTorrent clients on hundreds of Internet hosts, we show that on average TopBT can reduce about 25% download traffic while achieving a 15% faster download speed compared to several prevalent BT clients. TopBT has been widely used in the Internet by many users all over the world.

I. Introduction

Peer-to-peer (P2P) systems have been widely deployed and used to provide different services, such as file sharing, video streaming, and voice-over-IP. According to a late report by IPOQUE [1], P2P accounts for 73% of total Internet traffic. In particular, BitTorrent (BT), a P2P file sharing application, contributes 67% of this P2P traffic. BT gains an extreme user popularity for file sharing due to its inherent scalability and significantly reduced download time compared to the traditional client-server based downloading. Though BT has been a successful Internet application, it raises various challenges to network resource management, such as Internet Service Providers (ISPs), and affect other online applications. Often such a tremendous amount of P2P traffic has unnecessarily consumed Internet bandwidth that could have been used by other applications suffering from bandwidth shortage. For this reason, some ISPs have attempted to shape, deny, or suppress BT and general P2P traffic. For example, Comcast started to throttle P2P traffic in late 2007 to prevent P2P applications from significantly degrading the performance of other applications [8].

In major P2P systems, including BT, there is mutual-blindness between the application layer, known as overlay, and its network layer, or underlay. A few designs, such as Ono [7] and P4P [29], have been proposed to bridge the communications and interactions between these two layers, aiming to improve Internet bandwidth utilization and maintain user-perceived performance at the same time for such P2P systems. In these designs, a P2P client strives to connect to and download from other peers in network proximity, often in the same ISP. Although Ono and P4P have demonstrated the benefits gained by integrating network layer routing information into application design in real BT experiments, they have the following limitations that may significantly hinder their further deployment in the Internet and thus effectiveness from user perspectives.

First, these designs require feeds from major Internet infrastructures. Specifically, Ono relies on existing Content Distribution Networks (CDNs) to operate properly, imposing a significant amount of traffic on CDNs that might not be willing to offer such a free service constantly. On the other hand, P4P requires ISPs to provide network-layer information to applications, for which ISPs might not cooperate. In general, a private or enterprise network does not publicize its confidential internal network data due to security and privacy concerns. Second, these designs depend on large-scale deployments of their clients or interfaces. Particularly, Ono peers have to exchange CDN based coordinates to calculate network proximity. Given that the majority of BT clients are not Ono-based, in practice, an Ono client may not be able to accurately locate nearby non-Ono peers to establish connections. Similarly, if the majority of ISPs do not deploy P4P iTracker interfaces, P2P clients in those ISPs will not be able to retrieve the necessary information required for performance optimization. Third, these designs use metrics indirectly related to performance optimization objectives of the BT application. In Ono, peers measure their coordinates based on CDNs, which optimize for lots of metrics. These metrics do not reflect peer throughput that is critical to download time, as evidenced by our measurement results. On the other hand, P4P virtual topology information provided by ISPs might be different from traffic optimization metric of applications measured by routing hops. Finally, based on a trace analysis, a recent study [22] shows that ISP-friendly based BitTorrent, such as Ono and P4P, may have limited performance gain, and may reduce robustness by focusing on reducing traffic for a single ISP.

To address the above concerns, we have designed,

implemented, and evaluated an infrastructure-independent, topology-aware, and client-based BT system, called TopBT, which can significantly improve Internet resource utilization efficiency without degrading user downloading performance. The unique feature of TopBT lies in its comprehensive peer selection metric considering both the downloading speed and network topology with a candidate peer simultaneously. More specifically, a TopBT client launches lightweight pings or traceroute probes to its connected peers periodically, and maps connections to their corresponding link hops or Autonomous System (AS) hops. By passively monitoring connection transmission rates, a TopBT client always tries to unchoke peers that have low routing hops and provide high downloading rates. A TopBT client does not require feeds from ISPs or CDNs, nor requires other clients to be TopBT for it to achieve fast download time and reduce traffic. We have implemented the TopBT client based on Vuze, BitTornado, and LH-ABC, all are mainstream open-source BT clients written in Java and Python, respectively. We have released TopBT software for both Linux/Unix and Windows. Through distributed experiments on hundreds of PlanetLab and residential hosts, we observe that TopBT can reduce more than 25% traffic and download 15% faster with lightweight overhead when compared to a few popular BT clients.

TopBT has quickly attracted active attention in the BT community to leverage its effectiveness to improve the downloading efficiency of P2P systems. For example, the TopBT peer selection policy has been integrated into LH-ABC, a widely used native BitTorrent variant based on open source BitTornado with hundreds of thousands of accumulated downloads since 2006. So far, TopBT client has been widely used by increasingly more users worldwide, since it was made publicly available in August 2008. The quick and enthusiastic response from the P2P community further confirms that there is a strong demand to provide a client-based BT facility that can easily benefit users and Internet with topology-aware optimization. TopBT has also been independently evaluated and reported by FileShareFreak [10] – a professional website for "BitTorrent & P2P Tips and Information" as follows: "TopBT can not only reduce unnecessary BitTorrent traffic that clogs up the Internet, but it also downloads fast, even faster than μTorrent in our test cases."

The research and development of TopBT has made the following contributions.

1) We have developed a BT measurement framework, and collected BT file sharing data on hundreds of PlanetLab and residential hosts. With this data set, we have quantitatively analyzed and demonstrated that a large amount of Internet traffic is unnecessary.
2) By analyzing the collected BT workloads, we have shown that several typical network-level metrics, such as latency, when used for peer selection, would not be able to balance user downloading performance and the incurred network traffic.
3) We have designed a TopBT peer selection policy in which a TopBT client utilizes widely available network probing facilities to discover peers in proximity. Being infrastructure-independent, a single TopBT client can operate at user-level without requiring a large-scale deployment of other peers in same type to gain performance. TopBT aims at reducing the overall Internet traffic, thus gaining benefits from its global optimization and retaining the robustness of overlay topology.
4) We have implemented a new BitTorrent client software that has been demonstrated to improve user download time and reduce cross-ISP traffic simultaneously. The TopBT software has attracted a significant number of users all over the world. It has been integrated into Vuze, a mainstream open source BT client.
5) Fast download speed and high traffic-efficiency do not contradict each other in the TopBT environment. Our experiments show that both objectives can be achieved if a sufficient number of TopBT clients are deployed.

The remainder of the paper is organized as follows. In Section II, we sketch BT background and related work. We describe TopBT system design and implementation issues in Section III. In Section IV, we elaborate the TopBT peer selection policy. We evaluate TopBT performance in Section V. Finally, we make concluding remarks in Section VI.

II. Background and Related Work

BitTorrent (BT) is a widely used P2P based file sharing tool. In BitTorrent, a file is divided into multiple small pieces so that every client can exchange different pieces simultaneously, and thus significantly speed up the downloading process. The clients interested in a same file form a BitTorrent *swarm*, and the downloading of these clients are coordinated by a tracker site. In a swarm, peers with all file pieces are called *seeds*, and other non-seed peers are *leechers*. On bootstrapping, a peer connects to the tracker site based on the metadata in the torrent files to locate other peers. Once having established connections with other peers, a client can choose a subset of these peers to upload pieces to (i.e., *unchoked*), while blocking piece uploads to other peers (i.e., *choked*). Many popular BitTorrent variants [2], [4], [5], [17], [27] have been used for file distribution. Although these systems somewhat differ in their implementation details, almost all of them adopt random peer selection to establish and unchoke connections. Given its high scalability and productivity of BitTorrent, a number of studies have been focused on BT measurement, modeling, and incentives. A comprehensive study of BT can be found in [12]. Request distributions and statistical models of Internet media contents shared by BT and other tools have been studied in [13].

The significant amount of BT traffic on the Internet has caused ISPs to throttle, shape, or even block it. Targeting BT traffic reduction, researchers have proposed biased peer selection for BT systems by using network intelligence. Papers [3] and [15] suggested to select peers in the same ISP. The effectiveness of this approach is arguable considering that most BT peers are not in the same ISP [22].

Aside from BitTorrent, coordinate based [16], non-coordinate based [11], [28], and other locality-aware approaches [14], [18], [20] have been proposed for general P2P and distributed systems. However, all these methods are

designed to optimize network related performance factors such as latency, rather than download time and traffic reduction that we study in this work.

III. TopBT System Framework

To address the limitations and concerns of existing approaches, we strive toward an approach that can be infrastructure independent (particularly ISP independent) and operates independently to meet the optimization goal of reducing BT traffic across transmission paths while maintaining fast download speed. Our solution is TopBT, a topology-aware BT client that comprehensively considers connection rates and path topology when selecting peers to transmit data to/from.

A. TopBT Components

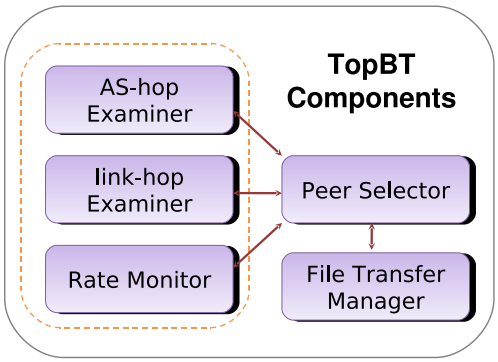


Fig. 1. TopBT structure and components.

TopBT consists of five components: *AS-hop examiner*, *link-hop examiner*, *rate monitor*, *peer selector*, and *file transfer manager*, as shown in Fig. 1. The AS-hop examiner and the link-hop examiner are responsible for discovering path proximity to connected peers; the rate monitor passively records download/upload throughput on each connection; the peer selector executes TopBT peer selection policy to choke/unchoke peers; and the file transfer manager is the component responsible for downloading/uploading file pieces from/to peers, managing connections, and writing data to I/O devices. We describe these components in details in the following sections.

B. Discovering Path Proximity

It is important for a TopBT client to be aware of peer proximities so that it can always download from close peers. To measure path proximity, a TopBT client takes the following steps.

Probing Connection Paths

A TopBT client probes its connected peers using network troubleshooting tools such as "ping" and "traceroute" (on Windows, "tracert"). Traceroute works by using the IP "time to live (TTL)" field, which specifies the maximum number of hops a packet may be forwarded by a router. In tracing a host, the traceroute program constructs a packet (usually UDP by default according to RFC 862, but Internet Control Message Protocol (ICMP) ECHO packets can be used in Windows) to an invalid port at the remote host, and sets its TTL field to 1. The first router that gets the packet will decrement its TTL field, check that it is zero, and return an ICMP TIME_EXCEEDED response to the originating host. Traceroute then prints out the IP address of the router that returned this message, creates a new packet with TTL 2, and so on, until the TIME_EXCEEDED message comes from a router when the maximum number of hops is reached, or an ICMP Destination Unreachable message is returned by the destination host.

On the other hand, in most systems, ping is usually implemented using the ICMP ECHO facility. A ping host sends an ICMP ECHO_REQUEST packet to the given destination. When no filtering device drops the packet and it arrives at the destination, the destination host creates an ICMP ECHO_REPLY packet with the same payload and sends it back as a response message.

Due to the fact that ping and traceroute use different types of packets, and routers along the path are capable of filtering each of these packets, we choose both tools for path topology discovery. In TopBT, its AS-hop and link-hop examiners periodically send traceroute and ping packets to those newly connected peers.

TCP Ping. Due to widely deployed firewalls and packet filters on routers and end hosts, a TopBT client frequently gets no response for traceroute and ping packets they send. To obtain a high response rate, a TopBT client sends TCP pings instead. That is, the client sends TCP SYN packets to peers, and extracts TTL values of subsequent SYN/ACK or RST packets.

Calculating Link-hops. Ping and TCP Ping results returned by remote peers contain TTL values when arriving at the original probing host. Depending on the operating systems of remote peers, the initial TTL values of a response packet can be set to different values. Typical and common initial TTL values are among 255 (most UNIX systems), 128 (Windows NT/2000/XP), 64 (Linux and Compaq Tru64), and 32 (Windows 95/98/ME) [26]. Early studies (e.g., [9]) have shown that 95% of Internet paths have link-hops of no more than 30, and our measurement confirms this result. Therefore, based on the TTL of the returning packet, we can infer the initial TTL of the packet, and thus the link-hops.

Calculating Autonomous System (AS) Hops

After obtaining the traceroute paths between a pair of connected peers, we use them to calculate AS-hops. We first build a prefix-AS mapping table by downloading up-to-date Border Gateway Protocol (BGP) routing table dumps from several public repositories including RoutesViews, RIPE NCC, and China CERNET, and merge all those BGP table entries. After filtering out traceroute paths containing IP-loops, for every non-looping traceroute path, we map each of its router interface IPs to its corresponding AS by looking up the prefix-AS table. Sometimes there are "*" hops on a traceroute path due to non-responding routers, and we manage to map them using several AS mapping techniques by considering previous and next ASes [19], [25]. The prefix-AS table is embedded into the AS-hop examiner, and can be refreshed once in several weeks from the server. Note that we build this table off-line in advance, and the client does not need to do any extra

calculation at runtime. The client simply uses this table to map IPs to their ASes. In other words, a client does not need to contact these public repositories at runtime, and thus, is independent of the infrastructure.

Asymmetric traceroute paths. The path from a TopBT host to a remote peer on which traceroute packets traverse, known as *forward path*, might differ from the *reverse path* through which the remote peer reaches this host. This is because routing tables on border gateway routers can dictate different paths due to their autonomous nature. To address this difference, two connected hosts can exchange AS-hop data with each other, so that a host can use *reverse path* proximity for its peer selection process.

MPLS networks. In large ISPs where Multi-Protocol Label Switching (MPLS) is deployed, Label Switch Routers (LSRs) exchange labels to decide forwarding paths for incoming packets. The routers in the MPLS core network are hidden from IP routing based traceroute and ping. Fortunately, as long as ingress and egress routers in the MPLS network appear on the traceroute path, the AS hops are not affected by MPLS. On the other hand, MPLS pings, when available on a TopBT host, can help calculate link hops to peers.

C. Monitoring Connection Rates

For every packet a TopBT client sends or receives, the client tracks the specific peer that the data go to or come from. Using a moving average method, in a time window that slides forward, the client counts the total bytes being transmitted and received, and divides the sums by the time window size to get connection download/upload rates. TopBT inherits the default configurable time window size used in the base BitTorrent software, which is usually at the scale of seconds.

D. TopBT Software and Status

We have developed both Windows version TopBT with installer and Linux version TopBT based on Vuze [2], LH-ABC [17], and BitTornado [5], three mainstream BitTorrent clients with millions of users. The initial version of TopBT was released to public in August 2008, and it has been optimized several times for better performance since then [1]. TopBT has attracted a significant number of active users from all over the world since its release, as shown in Fig. 2.

Fig. 2. TopBT usage geo-distribution (as in March, 2010).

IV. Exploring Peer Selection Policies

We use two factors, *amount of traffic* and *download time*, to characterize BT performance from the perspectives of both Internet and end users. Accumulated BT traffic, i.e., the total number of BT bytes that have accumulated on different network links and across ISP borders, reflects the stress to ISPs. While download time, i.e., the duration from BT user starts downloading to the time when the downloading completes, reflects the user satisfaction. Our objective is to design a strategic peer selection policy to reduce generated BT traffic while maintaining fast download speed.

In order to meet this goal, we need to deeply understand how BT traffic is transmitted, what factors affect user download time, and how we can combine reductions of both traffic and download time in our design. We first conducted a set of distributed experiments on native BT clients, i.e., BT clients that use the original BT peer selection policy, and analyzed their results.

A. Native BT Measurements for Peer Selection

We implemented a measurement framework called BT-Meter to conduct our experiments. BT-Meter is based on BitTornado, an open-source BitTorrent software written in Python. BitTornado uses the same peer selection policy as the original BT protocol, and we refer BitTornado as native BT. In all BT protocols, including both original one and its variants, file chunks, or pieces, are further divided into smaller slices. To measure native BT traffic and user download time, we developed modules to record BT connections and file states at the *slice level*.

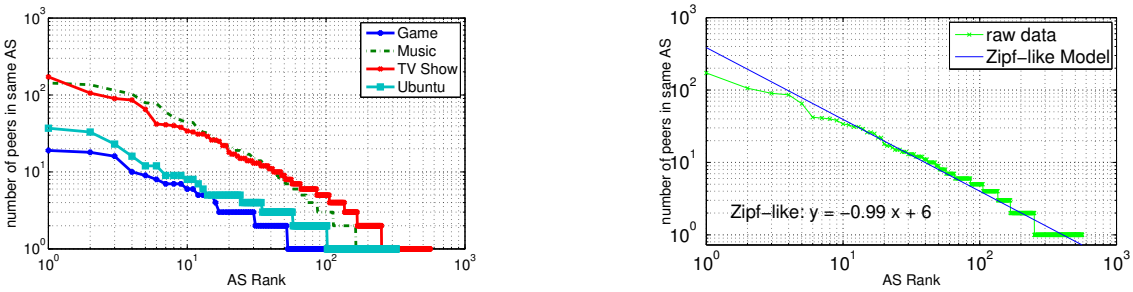
Using TopBT's components *AS-hop examiner* and *link-hop examiner*, we measure the AS-hops and link-hops for each connection. We also measure download/upload rates using TopBT's *Rate Monitor* component, as described in Section III-C. In addition, BT-Meter uses the native BT peer selection policy in its *peer selector* to select peers with higher download rates first. We develop a *slice monitor* component to periodically record BT context, such as the timestamp when the slice is completely downloaded, the number of connections of the host, and the other connected peers having the slice at the current moment. The *file transfer manager* remains the same for connection maintenance and file piece management.

name	size (bytes)	piece size (bytes)	peers
Game	179.0 M	262,144	510
Music	97.4 M	262,144	3,015
TV-show	348.6 M	262,144	3,821
Ubuntu	695.8 M	524,288	933

TABLE I
Torrent file summary.

To conduct measurements, we have chosen some hosts in universities and residential networks, and have deployed BT-Meter onto 106 PlanetLab (PL) [23] and residential hosts. These hosts are instructed to participate in different torrent swarms to download various files, which have different file

[1]Our TopBT client software and datasets are publicly available at http://topbt.cse.ohio-state.edu and http://sourceforge.net/projects/top-bt/



(a) # of peers in ASes.

(b) Zipf fit of # of peers in AS in TV.

Fig. 3. Swarm peers distribution in ASes.

sizes and different peer population. Table I gives a summary of the torrents. All files use a slice size of 16,384 (16K) bytes. The experiments were conducted in April, 2008, and we repeated once every day in a week, given that it took less than one day to complete on all hosts. Then on every host, we use the average of its values from the 7 runs to plot performance curves in figures.

B. Peer Selection Metric

The BT traffic is mainly decided by the routing hops of connected peers. The download time is determined by the file size and the aggregated download rate, which is the sum of all connections' download rates of the host. Since the file size is fixed, the file download time is thus affected by the number of connections as well as download rate of each connection. To design a peer selection policy that can balance both traffic and download time, we need to study all factors that can affect these two targets and derive an appropriate metric for peer selection.

We use two metrics, link-traffic and AS-traffic, to characterize the underlying traffic that could have been reduced. Considering that a file consists of n slices of equal size δ, we define *link-traffic* of an entire file transmission as the amount of network traffic traversing all IP links. Considering that these slices traverse $l_1, l_2, \ldots, l_n$ link hops, respectively, then the *link-traffic* is calculated as $\delta \sum_{i=1}^{n} l_i$. Similarly, we define *AS-traffic* of a file transmission as the total amount of network traffic across ASes (or ISPs) borders. Considering these n slices traverse $a_1, a_2, \ldots, a_n$ different ASes, respectively, we calculate the AS-traffic as $\delta \sum_{i=1}^{n} (a_i - 1)$. These calculations indicate that the link- or AS-traffic is equal to the mean link- or AS-hop times the file size. For a given file to download, the link- or AS-traffic is proportional to the mean link- or AS-hops, and mean link- or AS-hop reduction reflects link- or AS-traffic reduction. Note that link-traffic characterizes the average link traffic stress, and AS-traffic characterizes the cross-AS traffic. Because most ISPs consist of only one AS, and they are charged based on cross-ISP traffic, AS-traffic reduction usually means less billing to them.

We first study two peer selection policies solely based on traffic related factors.

Selecting Peers in the Same AS

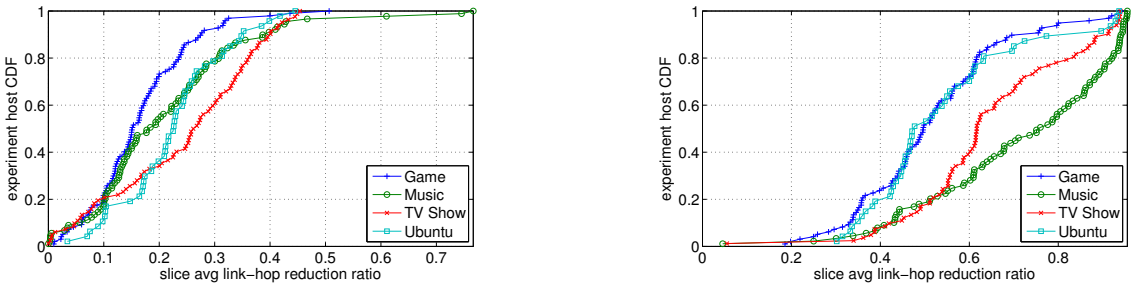
One way to select close peers is to select peers from the same AS, as suggested by a few studies [3], [15]. Fig. 3(a) shows the number of swarming peers in different ASes for the four torrents, and Fig. 3(b) shows a Zipf fitting for the TV show torrent. In these figures, the x-axis is the AS index in descending order of peer number, and the y-axis is the actual number of peers. These results indicate that the number of peers in different ASes roughly follows a Zipf distribution with α close to 1. For any given client, although there are peers in its own AS, most other peers reside in different ASes. Therefore, the client's own AS does not have enough peers to be selected for fast downloading in most cases.

Selecting Peers with Lowest Hops

To study BT traffic upper bound, we quantitatively study what if a peer in downloading these files had selected a nearby peer.

Fig. 4(a) shows the mean link-hop reduction of file transmission processes in 4 different swarms. Note that all hosts here have successfully completed their downloadings. These 4 swarms are independent, and their durations do not overlap with each other. In this figure, the x axis is slice-level mean link-hop reduction ratio, and the y axis is the cumulative distribution function (CDF) of the experiment hosts. This is calculated based on the following procedure: when peer p selects a peer to download from (unchoke), it selects from currently connected peers. The selected peer is not necessarily the best peer that has the shortest link-hop distance from peer p. Thus, by assuming the best peer will eventually have the slice wanted by this peer, the link-hop reduction percentage is calculated based on the difference between these two, divided by the truly incurred link-hops. This figure shows that using the shortest link-hop distance as the peer selection policy, half of the hosts can achieve more than 20% mean link-hop reduction for all four swarms.

If a peer does not constrain its selection among all connected peers, but to all the available peers, the reduction could be more significant. Fig. 4(b) thus shows the mean link-hop reduction when a peer is allowed to do so. The total available peer list is the combination of peer lists of all nodes in the same swarm. If we compare Fig. 4(b) with Fig. 4(a), we observe a much larger mean link-hop reduction across all torrents: half of the hosts can achieve more than 50% mean link-hop reduction. The average link-hop reduction per node is about 60%. Note that the amount of reduction in this figure indicates the upper bound. In practice, when the number of peers in the swarm is very large, and link-hops between these peers are very diverse, the traffic reduction could approach this



(a) Link-hop reduction with only connected peers.

(b) Link-hop reduction with all available peers.

Fig. 4. Link-hop reduction across the experiment hosts.

bound.

Fig. 4(a) and Fig. 4(b) show that in existing BT systems, based on link-hops, a significant amount of network traffic could have been reduced if the peer has carefully selected peers to download from.

In addition to link-hops, we also use AS-hops as another metric to study how much AS-traffic could have been reduced. We found that, similar to the link-hop reduction results, about half of the hosts can achieve more than 25% mean AS-hop reduction using only connected peers. If using all available peer list, half of the hosts can achieve more than 70% AS-hop reduction. The average AS-hop reduction per node is about 74%. We observe from our collected data that more than 50% of connections have AS-hops below 5. Thus, in practice, it is very likely for a host to find peers within the same AS or in nearby ASes that have less than 5 AS-hops.

The Comprehensive Unchoking Algorithm in TopBT

We have conducted a wide range of experiments to measure and understand the relationship between transmission distance (the number of routing hops) and transmission rates, and documented detailed results in [24]. In our measurement results, we observe no strong correlations between the routing hops (link- or AS-hops) and the transmission rate (downloading or uploading rate) for a connection, as shown in Fig. 5.

Fig. 5 plots performance curves from more than 9000 P2P connections collected in our experiments. We first sort the connections by their number of link hops, and then sort connections with the same number of hops by their downloading rates. The x-axis is the rank number of each connection; the left y-axis is the number of link hops, and the right y-axis is the downloading rate. For example, Connection #3000 and Connection #3800 have the same number of hops, since they fall onto the same segment when projected vertically. However, they have very different downloading rates.

Fig. 5 implies that a downloading-rate-driven peer selection policy in native BT is not necessarily able to find a desirable peer with a high uploading rate and small routing hops. Similarly, traffic oriented policies may not find close peers that also have high uploading rates. For a BT peer in a swarm, there may exist multiple peers with similar downloading rates but significantly different routing hops to this peer.

Motivated by our Internet measurement results, we propose a comprehensive peer selection algorithm based on the collected information of both traffic and transmission rate. In this algorithm, peers are placed into four categories according to their hops and downloading rates. The categories contaion peers that are: fast&close, fast&far, slow&close and slow&far respectively.

Fig. 6 shows how peers are categorized. Its x-axis is 1/(# of Hops) and y-axis is the downloading rate a local client gets from the peer. Peers that are both fast and close (upper-right corner in Fig. 6) are the best candidates in TopBT, while slow and far peers (lower-left corner in Fig. 6) are the worst.

TopBT unchokes peers within the "fast&close" category. Due to the fixed size of the active set, the "fast&close" category may contain more peers than required. In this case, we select the most qualified peers in this category. It is also possible that the "fast&close" category doesn't have enough peers. When this happens, it usually means that the P2P resource is not sufficient, and users may experience performance degradation if they still insist on reducing traffic. In this case, besides peers in the "fast&close" category, we choose the fastest peers in the "fast&far" category (upper-left corner in Fig. 6) to fill the rest of the unchoking slots.

Currently, the unchoking algorithm does not consider the uploading rate at which data is transmitted from local client to a remote peer, as is done in [21], because this will cause connection fluctuations [6].

C. TopBT Peer Selection Policy

In TopBT, there are several places to incorporate topology-awareness for effective peer selection: tracker returned peer list, initial connection establishment, connection replacement, and unchoking mechanism. As a standard terminology in BT community, "unchoke" a peer means the client allows data to flow to that peer. A TopBT client always aggressively retrieves peer lists from the tracker site, so that it can get a large set of peers. In this section, we focus on the TopBT unchoking mechanism. As a principle can be applied to the other places as well.

Unchoking Mechanism in TopBT

A TopBT client unchokes its connected peers round by round. The round duration is configurable, and each round usually lasts several minutes. At the end of each unchoking round, a TopBT client computes hops and the transmission rate of each connected peer. Once the best peers are determined, the client unchokes them simultaneously.

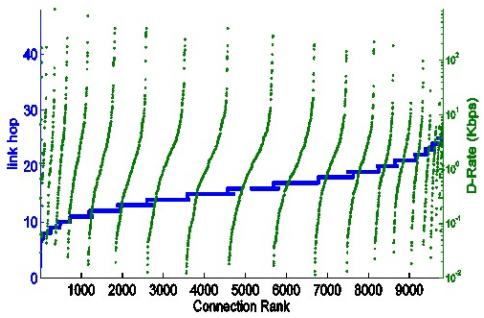


Fig. 5. No strong correlation between link hop and downloading rate.

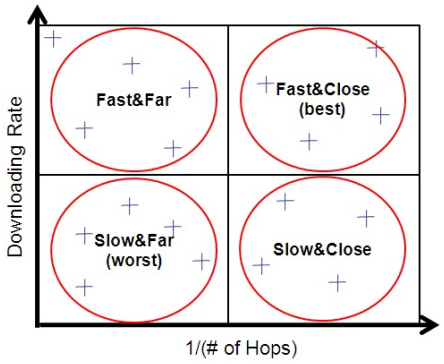


Fig. 6. Peers are placed into 4 categories in TopBT.

At the starting phase before data transmission, a TopBT client initializes the downloading rate of every connection to its download capacity equal split, i.e., download capacity divided by the number of connections;

Once connections are established, in each unchoking round, the TopBT client can passively monitor downloading rates for mutually unchoked peers, and measure link- and as-hops using the link-hop examiner and the AS-hop examiner.

Handling Non-responding Peers

For those peers not responding to pings or traceroute probes, we use the average routing hops of responding peers as the estimation of their hops. Although it may possibly unchoke faraway peers that provide moderate downloading rates, those close and fast responding peers are guaranteed to be unchoked. An alternative is to divide connected peers into two groups, i.e., responding group and non-responding group. The client then allocates upload caps to the two groups based on their group sizes. For the responding group, it uses the comprehensive metric to sort peers; while for the non-responding group, it uses the downloading rates to sort.

Limitations With Tracker-Side Approach

Although one may argue to put the peer selection at the tracker side, tracker is unable to measure the routing hops and real-time connection rates between any two connected peers. Thus, the tracker-side approach is difficult to implement in practice.

D. Benefits Gained From TopBT Deployment

Internet, including ISPs, can significantly benefit from a wide deployment of TopBT clients so that both cross-ISP traffic and average link stress can be reduced. From end users' perspective, download time is the most concern, i.e., how fast a file can be downloaded. Our experiments in the next section show that TopBT can achieve comparable or even better download speed than very fast BT clients, such as BitTyrant, after a sufficient number of TopBT clients have been deployed.

V. Experiments and Evaluation

To evaluate TopBT's performance, we conducted several large scale distributed experiments and compared against *native BT* and *BitTyrant* [21]. To sketch, a native BT client unchokes peers with highest download rates until its upload rate cap is reached, and a BitTyrant client unchokes peers with highest download-rate/upload-demand ratios until the upload rate cap is reached.

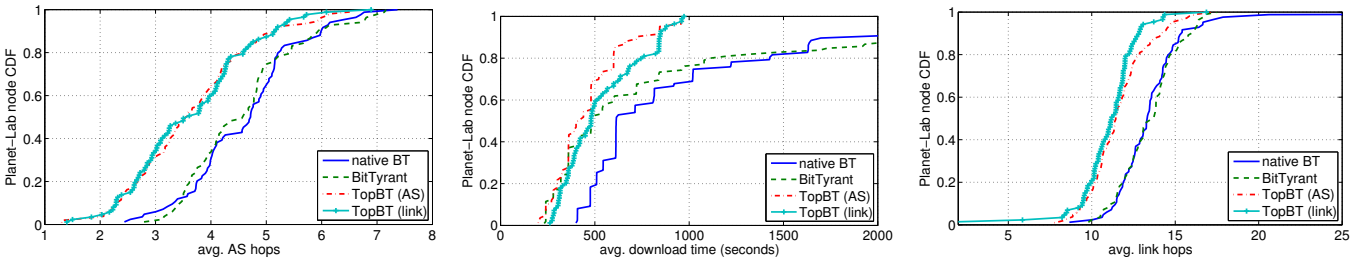
We deployed native BT, BitTyrant, and TopBT clients onto hundreds of PL hosts on the Internet. These PL hosts participated in downloading legal media files using an existing popular torrent and a private torrent. For the popular torrent, most peers in the torrent swarm are non-PL hosts, while for the private torrent, the swarm contains only the hosts running our selected BT client. In this way, we can study the scenarios of mixed BT clients as well as the single BT client. We started native BT, BitTyrant, and TopBT in random order: when one completes, we randomly picked the next to start so that our results do not favor any of the three clients, given that a prior run can help proliferate the torrent pieces in the system, and thus possibly improve the performance of later runs. The PlanetLab experiments were repeated once every day from May 20 to May 27, 2008. We also conduct experiments on 14 residential hosts and repeat experiments 7 times during one week, using the same approach as our PlanetLab experiments. The mean values are used in the following report.

In our experiments, most of nodes can finish downloading in two hours, which is fast enough compared to the average download time of more than one day for all Internet hosts downloading the same torrent, as reported in the torrent tracker site. We think this is relatively short and hope there is no significant change on the Internet in the meantime.

Our performance metrics include download time, average AS-hop, and average link-hop. The average AS-hop (link-hop) is computed as the ratio of AS-traffic (link-traffic) over the file size. We have also studied the effects of other factors relevant to these performance metrics, including upload capacity, ratio of seeds and leechers, node probing delay, and non-responding peers. We evaluated TopBT overhead in terms of the amount of probing traffic it triggers.

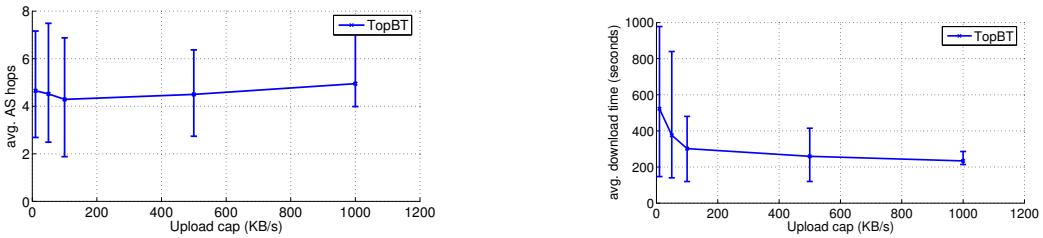
Traffic and Download Time

Fig. 7(a) shows that among the 100+ PL node experiments, link-hop and AS-hop based TopBT always achieves a lower number of AS-hops. Compared to native BT and BitTyrant, 30% TopBT PL nodes have 25% less number of AS hops. Given the large size of the BT downloading file, it implies that TopBT can reduce a significant amount of Internet traffic.



(a) Average AS-hop (b) Download time (c) Average link-hop

Fig. 7. TopBT experiment results.



(a) Average AS-hop (b) Download time

Fig. 8. Real deployment results on PL nodes under upload capacity constraint (TopBT based on AS-hop).

Link-hop based TopBT can achieve a slightly lower number of AS-hops than AS-hop based TopBT. This may be because AS-hops cannot be obtained for a large fraction of peers due to the fact that routers filtering traceroute packets, and the client cannot use the comprehensive metric to unchoke them.

For the download time, Fig. 7(b) illustrates that both TopBT and BitTyrant finish downloading about 15% faster than that of native BT. AS-hop based TopBT can download slightly faster than link-hop based TopBT, and TopBT's download time on all PL nodes is even slightly shorter than that of BitTyrant. We believe the observed superior TopBT download performance is contributed by our system environment where a sufficient number of TopBT clients are deployed, such that unnecessary traffic is minimized to benefit download time of each peer. In contrast, if every user makes aggressive peer selection operations, the accumulated bandwidth can be over-demanded to eventually harm the average download performance among the peers.

Again, for average link hop results, link-hop and AS-hop based TopBT are always much lower than native BT and BitTyrant, as shown in Fig. 7(c). This large difference is caused by topology-unawareness of native BT and BitTyrant. Link-hop based TopBT has a slightly lower link-hops than that of AS-hop based TopBT, due to its accurate counting of routing hops.

Effects of Upload Capacity

Fig. 8(a) shows that the upload cap does not affect the average AS hops, while Fig. 8(b) shows that a higher upload cap yields a shorter download time. When the upload cap is under 200K bytes/s, this effect is more pronounced: the average download time decreases quickly from 530 seconds to 370 seconds, roughly 50% faster with a larger upload cap.

Effects of Seeds and Leechers

Fig. 9 shows that a higher seed/leecher ratio results in a lower download time for TopBT. The download time drops from 1980 seconds to 833 seconds when the seed/leecher ratio increases from 0.52 to 1.64. This is because seeds do not consume the client's upload bandwidth, and thus, can increase the client download rate.

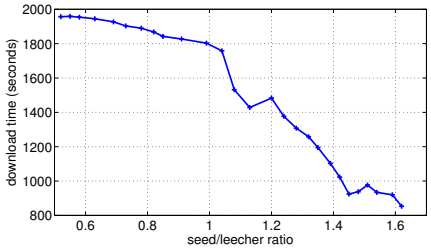


Fig. 9. Seed/leecher ratio effect on TopBT download time.

Effects of Hop Probing Delay

type	avg	min	max
tcp ping	0.58 sec	0.13 sec	7.24 sec
traceroute	40 sec	18 sec	2 min

TABLE II
TOPBT CLIENT PROBING DELAY.

Table II shows that the TopBT probing process in each round only takes seconds to finish. On average, TCP ping based TopBT only takes about half a second to measure link-hops, and traceroute based TopBT takes a longer time, about 40 seconds to complete probing. The probing process runs in the background. Before any useful routing hop values are measured, TopBT uses an unchoking mechanism similar to BitTyrant. Therefore, the probing delay in TopBT does not

affect its download time.

Non-responding Peers

With TCP ping, a TopBT client can measure link-hops for about 95% peers, since the TopBT implementation utilizes the listening or connection ports of remote peers for probing, which bypasses most firewalls and NATs that regular ping cannot bypass. While with traceroute, TopBT client can measure AS-hops for only about 40% to 55% peers. Comprehensively considering other factors, link-hop based TopBT is a better choice than AS-hop based TopBT.

TopBT Overhead

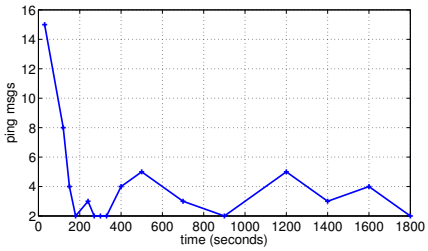


Fig. 10. TopBT ping messages generated during downloading.

Fig. 10 shows that a TopBT only generates a peak number of ping messages at the start during its unchoking period (<16 in 5 minutes), and that number quickly drops to a few (2 in 5 minutes). For traceroute, the number is higher, as each traceroute message triggers a sequence of ICMP packets. It is about 20 times more than the number shown in the figure. But overall, TopBT is light-weight in terms of extra probing messages generated.

VI. Conclusion

The BitTorrent-based file downloading consumes a huge portion of Internet bandwidth now. However, the current BT applications have over-utilized too much the Internet resource because of the lack of communications between the overlay and the underlay. The issue on how to reduce BT traffic without affecting user perceived download time remains challenging. To address this challenge, we have designed, implemented, and evaluated TopBT, a topology-aware and infrastructure-independent BitTorrent client. TopBT is based on open-source Vuze, BitTornado, and LH-ABC. We show that TopBT can retain low download time, and can reduce up to 25% induced Internet traffic, compared with several other representative BT clients. TopBT has been integrated into LH-ABC, a mainstream BT client, and has been widely used all over the world. We are currently further optimizing our comprehensive peer selection metric and relevant metrics, in order to continue to increasing the usage scope of TopBT for the benefits of both Internet and BT users.

VII. Acknowledgement

We thank annonymous referees for their appreciation of TopBT and their insightful and constructive comments. This work has been supported in part by U.S. NSF under grants CNS-0509054, CNS-0509061, CNS-0621629, CNS-0621631, CNS-0721516, and CNS-0746649, and by U.S. AFOSR under grant FA9550-09-1-0071. We thank Yin Huai for his effort in the maintenance of TopBT.

References

[1] Internet Study 2007. http://www.ipoque.com/resources/internet-studies/internet-study-2007.
[2] Azureus. http://www.azureus.com.
[3] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (IEEE ICDCS'06)*, Lisbon, Portugal, July 2006.
[4] BitComet. http://www.bitcomet.com.
[5] BitTornado. http://www.bittornado.com.
[6] D. Carra, G. Neglia, and P. Michiardi. On the Impact of Greedy Strategies in BitTorrent Networks: The Case of BitTyrant. In *Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, RWTH Aachen University, September 2008.
[7] D. R. Choffnes and F. E. Bustamante. Taming the Torrent. In *ACM SIGCOMM'08*, Seattle, WA, USA, August 2008.
[8] Comcast and BitTorrent Form Collaboration. Associated Press, October 2007.
[9] A. Fei, G. Pei, R. Liu, and L. Zhang. Measurements on delay and hop-count of the internet. In *IEEE GLOBECOM'98*, Sydney, Australia, November 1998.
[10] FileShareShark TopBT Review. http://filesharefreak.com/2009/01/23/topbt-on-vuze-a-topology-aware-bittorrent-client/.
[11] M. J. Freedman, K. Lakshminarayanan, and D. Maziéres. Oasis: Anycast for any service. In *USENIX NSDI'06*, San Jose, CA, USA, May 2006.
[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, Analysis, and Modeling of BitTorrent-like Systems. In *Proceedings of the 2005 USENIX International Measurement Conference (USENIX IMC'05)*, Berkeley, CA, USA, October 2005.
[13] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang. The Stretched Exponential Distribution of Internet Media Access Patterns. In *Proceedings of the 27th Annual ACM Symposium on Principles of Distributed Computing*, Toronto, Canada, August 2008.
[14] S. Horovitz and D. Dolev. Liteload: Content unaware routing for localizing p2p protocols. In *Proceedings of the 5th International Workshop on Hot Topics in Peer-to-Peer Systems (IEEE Hot-P2P'08)*, Miami, FL, USA, April 2008.
[15] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet Service Providers Fear Peer-Assisted Content Distribution? In *Proceedings of the 2005 USENIX International Measurement Conference (USENIX IMC'05)*, Berkeley, CA, USA, October 2005.
[16] J. Ledlie, P. Gardner, and M. Seltzer. Network coordinates in the wild. In *USENIX NSDI'07*, Cambridge, MA, USA, April 2007.
[17] LH-ABC: Enhanced Multi Platform BitTorrent Client. http://code.google.com/p/lh-abc/.
[18] J. Li. Locality aware peer assisted delivery: the way to scale internet video to the world. In *Proceedings of the 16th International Packet Video Workshop (IEEE PV'07)*, Laussane, Switzerland, November 2007.
[19] Z. Mao, J. Rexford, J. Wang, and R. Katz. Towards an Accurate AS-level Traceroute Tool. In *ACM SIGCOMM'03*, Karlsruhe,Germany, August 2003.
[20] R. Pereira, T. Vasques, and R. Rodrigues. Adaptive Search Radius - Lowering Internet P2P File-Sharing Traffic through Self-Restraint. In *Proceedings of the 6th International Symposium on Network Computing and Applications (IEEE NCA'07)*, Cambridge, MA, USA, July 2007.
[21] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *USENIX NSDI'07*, Cambridge, MA, USA, April 2007.
[22] M. Piatek, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson. Pitfalls for ISP-friendly P2P design. In *The Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, New York City, NY, USA, October 2009.
[23] PlanetLab. http://www.planet-lab.org.
[24] S. Ren. Analysis, design, and implementation of topology-aware overlay systems on Internet. PhD Thesis in Computer Science and Engineering, The Ohio State University, March 2009.
[25] S. Ren, L. Guo, and X. Zhang. ASAP: an AS-Aware Peer-Relay Protocol for High Quality VoIP. In *IEEE ICDCS'06*, Lisboa, Portugal, July 2006.
[26] Default Initial TTL Values. http://members.cox.net/∼ndav1/self_published/.
[27] uTorrent. http://www.utorrent.com.
[28] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: a lightweight network location service without virtual coordinates. In *ACM SIGCOMM'05*, Philadelphia, Pennsylvania, USA, 2005.
[29] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Provider Portal for Applications. In *ACM SIGCOMM'08*, Seattle, WA, USA, August 2008.