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, such as Ono and P4P, have been recently proposed and deployed 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 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 save about 25% download traffic while achieving a 15% faster download speed compared to several prevalent BT clients. TopBT has attracted a large number of downloads for a wide usage across 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 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 both Internet Service Providers (ISPs) and other online applications. Often such a tremendous amount of P2P traffic has unnecessarily consumed precious Internet bandwidth that could have been used by other applications suffering from bandwidth shortage. For this reason, 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].

Most recent studies show that 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 [32], 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 limits that can significantly hinder their deployment 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.

In this paper, we design, implement, and evaluate 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 ping 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 with low routing hops, high download rates, and low reciprocal upload rates at which those peers do not choke the client [25]. 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 save 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 release TopBT software for both Linux/Unix and Windows. Through distributed experiments on hundreds of PlanetLab and residential hosts, we observe that TopBT can save more than 25% traffic and download 15% faster with lightweight overhead when compared to a few popular BT protocols.

TopBT has quickly attracted special 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 Vuze, 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 attracted a significant number of downloads since it was made publicly available in August 2008, being actively used in various geographical locations worldwide. 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 [11] – a professional website for "BitTorrent & P2P Tips and Information" as follows: "TopBT can not only save unnecessary BitTorrent traffic that clogs up the Internet, but also download fast, even faster than μTorrent in our test cases." Interested readers may download TopBT to test its performance at http://topbt.cse.ohio-state.edu.

The contributions of our work are fourfold.

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 bandwidth could have been wasted.
2) By analyzing the collected BT workload, 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 large-scale deployment of its same types to immediately gain performance.
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 downloads and runs, and has been integrated into Vuze, a mainstream open source BT client.

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 one of the most popular 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], [5], [6], [19], [30] have been widely used for file distribution. Although these systems slightly differ in their implementation details, almost all of them adopt random peer selection to establish and unchoke connections. Given the super scalability of BitTorrent, a number of studies have been focused on BT measurement [3], [9], [13], [15], [27], modeling [18], [22], [28], [33], and incentives [23], [25], etc.

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. Bindal et al. [4] and Karagiannis et al. [16] suggested the ideas of selecting peers in the same ISP. The effectiveness of these same-ISP approaches is arguable given that most BT peers are not in the same ISP. With more complicated knowledge from either CDNs or ISPs, Ono [7] and P4P [32] recommend to select peers based on network proximity. Both of these approaches require existing infrastructure support, wide deployment of their new clients or interfaces, and accurate proximity feeds.

Aside from BitTorrent, coordinate based [17], non-coordinate based [12], [31], and other locality-aware approaches [14], [20], [24] have been proposed for general P2P and distributed systems. However, all these methods are designed to optimize other metrics such as latency, rather than download time and traffic saving that we study in this work.

III. TopBT System Framework

To address the limits of existing approaches such as P4P and Ono, we strive toward an approach that can be infrastructure 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

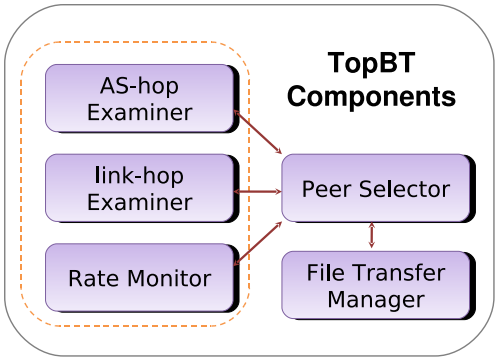


Fig. 1. TopBT structure and components.

TopBT consists of *AS-hop examiner*, *link-hop examiner*, *rate monitor*, *peer selector*, and *file transfer manager* components, 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 with 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 steps as shown below.

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 ping 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) [29]. Early studies (e.g., [10]) 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 [21]. 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, the 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 overcome 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 ping, 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 develop both Windows version TopBT with installer and Linux version TopBT based on Vuze [2], LH-ABC [19], and BitTornado [6], 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 downloads from all over the world since its release.

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

IV. Exploring Peer Selection Policies

We use two factors, *traffic* and *download time*, to characterize BT performance from the perspectives of both ISPs and end users. Accumulated BT traffic, i.e., the total BT bytes that have accumulated on different network links and across ISP borders, reflects ISP stress. While download time, i.e., the duration from BT user starts downloading to the time when the downloading completes, reflects 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 both traffic and download time in our design. We first conduct a set of distributed experiments on native BT clients, i.e., BT clients that use the original BT peer selection policy, and analyze 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 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) [26] and residential hosts. These hosts are instructed to participate in different torrent

swarms to download various files, which have different file 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 draw 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 values and derive an appropriate metric for peer selection.

1) Traffic Related Factors : We use two metrics, link-traffic and AS-traffic, to characterize the underlying traffic that could have been saved. Suppose the 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. Suppose 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. Suppose these n slices traverse $a_1, a_2, \ldots, a_n$ different ASes, respectively, then the AS-traffic is calculated 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 saving reflects link- or AS-traffic saving. 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 saving usually means less billing to them.

We first study two peer selection policies based solely 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 [4], [16]. Fig. 2(a) shows the number of swarming peers in different ASes for the four torrents, and Fig. 2(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. 3(a) shows the mean link-hop savings of file transmission processes in 4 different swarms. Note that all hosts here have successfully completed their downloading. 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 saving ratio, and the y axis is the 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 necessary 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 saving 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 saving for all four swarms.

If a peer p does not constrain its selection among all connected peers, but to all the available peers, the saving could be more significant. Fig. 3(b) thus shows the mean link-hop saving 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. 3(b) to Fig. 3(a), we observe a much larger mean link-hop saving across all torrents: half of the hosts can achieve more than 50% mean link-hop saving. The average link-hop saving per node is about 60%. Note that the amount of saving 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 saving could approach this bound.

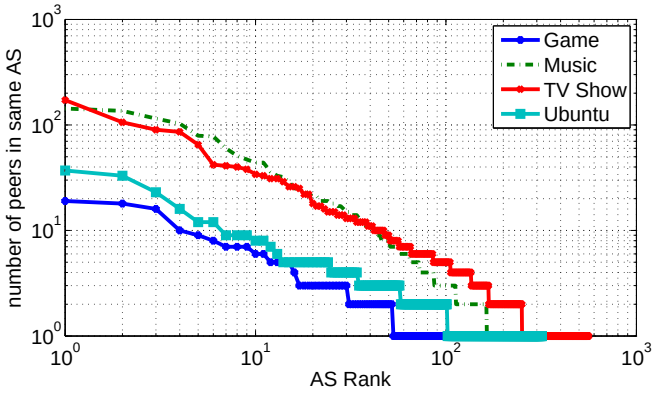
Fig. 3(a) and Fig. 3(b) show that in existing BT systems, based on link-hops, a significant amount of network traffic could have been reduced if the peer has carefully selected peers to download from.

In addition to link-hops, we also use AS-hops as another metric to study how much AS-traffic could have been saved. We found that, similar to the link-hop saving results, about half of the hosts can achieve more than 25% mean AS-hop saving using only connected peers. If using all available peer list, half of the hosts can achieve more than 70% AS-hop saving. The average AS-hop saving 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, there is a good chance for a host to find peers within the same AS or in nearby ASes that have less than 5 AS-hops.

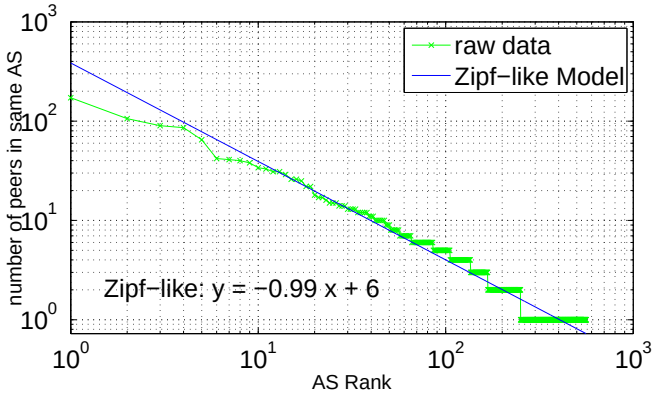
AS-Hop and Link-Hop Relationship

It is necessary to study the relationship between AS-hops and link-hops before we can answer the question whether one of these two factors can substitute the other. Fig. 4 shows the corresponding AS-hops and link-hops for different peer pairs for the TV torrent.

The x axis is the rank of these peer pairs in ascending order of AS-hops, and the y axis represents the corresponding

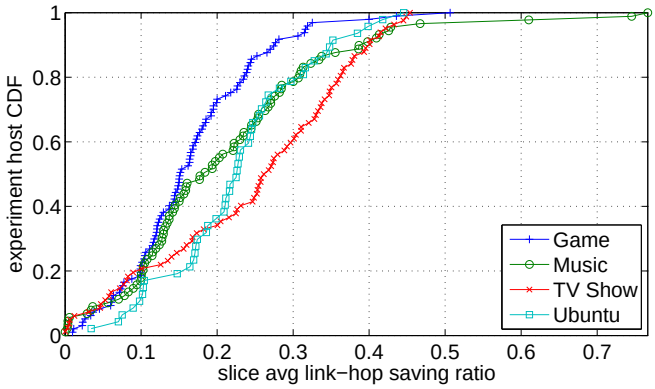


(a) # of peers in ASes.

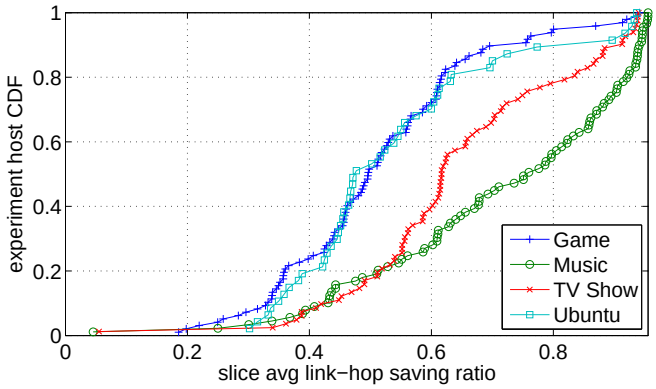


(b) Zipf fit of # of peers in AS in TV.

Fig. 2. Swarm peers distribution in ASes.



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



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

Fig. 3. Link-hop saving across the experiment hosts.

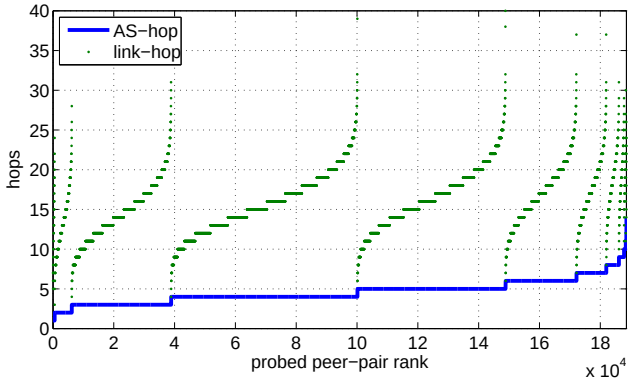


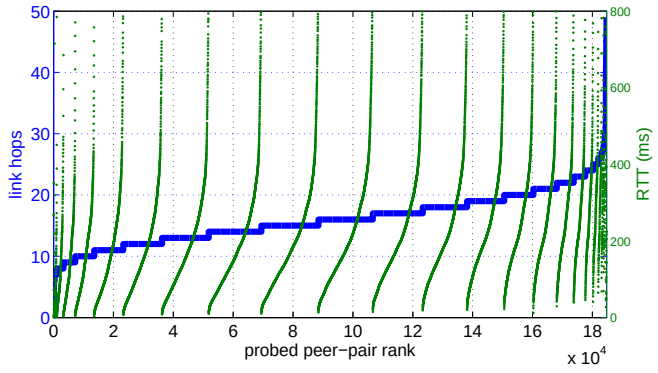
Fig. 4. AS-hop and link-hop relationship.

routing hops. The solid line is for AS-hops, and the dotted line is for link-hops. The figure shows that for each distinct value of AS-hops, the distribution of link-hops is similar and nearly in the same range (from AS-hops to 30). We can also observe that the median of link-hops for each distinct AS-hop increases as the AS-hop increases from 0 to 15. This indicates that link-hops are positively correlated to AS-hops with a correlation coefficient of 0.27 that we calculated from the data, and link-hops are always larger than AS-hops.

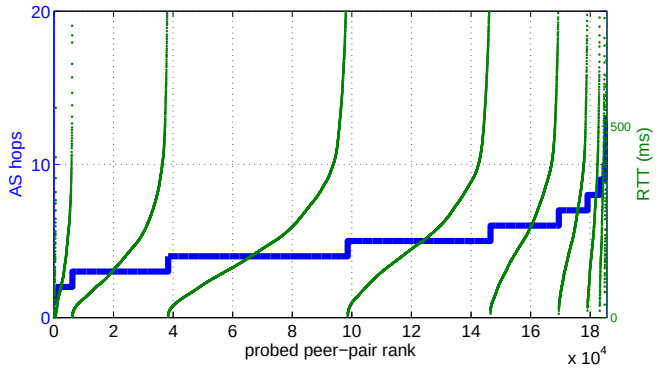
Routing Hop and RTT Relationship

Our measurement results so far indicate that link-/AS-hop, if considered in peer selection, could have reduced Internet traffic for BT downloading. However, hop is a metric that is more difficult to measure and less widely used than some other network metrics, such as Round-Trip Time (RTT). RTT is widely used to estimate network distance between end hosts. And thus, it is necessary for us to study whether RTT is a good indicator of routing hops between peers.

Fig. 5(a) shows the relationship between link-hop and RTT for the same set of peer pairs for the TV torrent. The x axis is the rank of peer pairs in ascending order of link-hops. The left y axis shows the number of link-hops for the thick horizontal ladder line, while the right y axis shows the corresponding RTT for the multiple parallel vertical lines. This figure shows that for each distinct value of link-hops, the distribution of RTT of corresponding peer pairs is also similar to each other, nearly in the same range (0-800 ms). We observe that when the link-hop increases, the median of RTTs increases as well, though not to a significant degree. It implies that there is

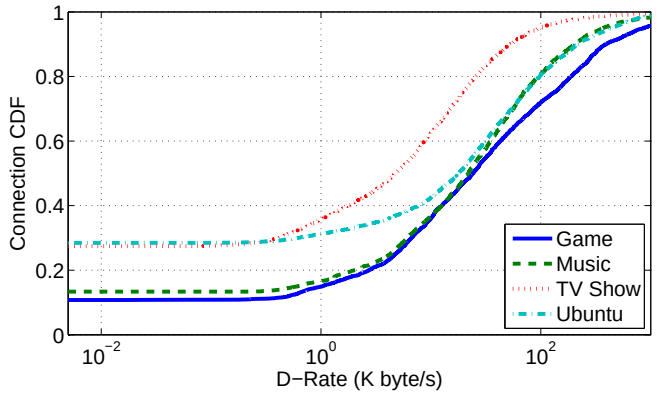


(a) Link-hop and RTT.

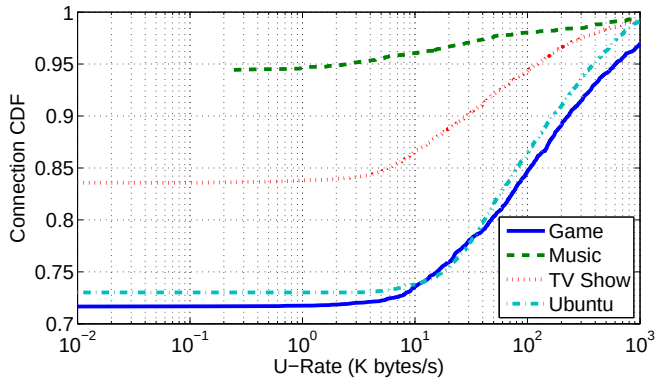


(b) AS-hop and RTT.

Fig. 5. Routing hops and RTT relationship for TV.



(a) Download rate.



(b) Upload rate.

Fig. 6. Connection download and upload rate distribution.

small positive correlation between link-hop and RTT, with a correlation coefficient of 0.22 that we calculated. We have also analyzed AS-hop and RTT relationship. Figure 5(b) shows that for each distinct value of AS-hops, the distribution of RTT between the corresponding peer pairs is similar to each other and nearly in the same range (0-800 ms). From this figure, we can observe small positive correlation between AS-hop and RTT as well, with a correlation coefficient of 0.17.

2) Download Time Related Factors: In practice, a BT user always intends to set a small upload capacity constraint but a large download capacity constraint, which we short as upload cap and download cap. Even if a BT user does not set upload and download caps, in most residential networks, the upload capacity is much smaller than the download capacity due to ISP's asymmetric bandwidth allocation. Therefore, a BT user's uploading is often capped by the upload capacity (or available upload bandwidth). In either cases, there exists an upload cap constraint for a BT user. Since in BT, the "tit-for-tat" mechanism is used, the cap on the uploading will finally affect the downloading. Having observed the upload capacity constraints in BT downloading, Piatek et al. [25] propose to utilize both download rate and upload rate to optimize download time. With the scarce upload capacity, we want to use the minimal upload bandwidth to gain maximal download rate. For this reason, we study download rate and upload rate, and investigate their effects on the file download time.

Connection Rates

Fig. 6(a) shows the download rate distribution of different connections for each torrent. The median of active download rates is between 10K and 40K bytes/s. The rationale behind this small value is that most peers are sharing the Internet uploading link among tens of active TCP connections so that each connection can only get a small fraction of the total uploading bandwidth (around 64K bytes/s for most Cable or DSL users). Fig. 6(b) shows the corresponding upload rate distribution. As we can observe, most of the connections do not actively upload. This is probably due to the fact that most experiment hosts have good Internet connectivity, and thus can download from seeds for most of time since the seeds are in favor of fast downloaders.

Ratio of Connection Rates

Given the file download time is affected by both the download and upload rates, the ratio of download rate d over upload rate u could be used for peer selection in order to optimize download time. The larger the ratio, the higher the download rate (with the same unit of upload rate). We study the distribution of such a ratio of all connections, as shown

in Fig. 8. In this figure, we only draw leecher connections, because seed connections have an upload rate u of 0, which results in ∞ of d/u. Fig. 8 shows that only a small fraction of all connections have a high d/u ratio: less than 20% of all connections have d/u over 1.4. Furthermore, this figure shows that a high percentage ($> 60\%$) of all peers in the swarm are selfish, with d/u less than 1 across all four swarms. Less than 2% of all connections have d/u smaller than 0.01, while less than 1% of all connections have d/u larger than 100.

3) Comprehensive Metric: **Rates and Routing Hops**

As discussed previously, a good peer selection strategy should consider both download time and network traffic. Thus, it is important to understand whether and how routing hop affects the download rate between two end hosts. Fig. 7(a) presents the number of link-level hops (left y axis) and the corresponding download rates (right y axis) for each BT probing connection in the TV swarm, recorded by client link-hop examiner and rate meter. The x axis presents the rank of these connections in ascending order of link-hops. As shown in the figure, for each distinct value of link hops, the distribution of download rates of corresponding BT probing connections has a similar range (10^{-2}-10^{3} Kbps), and the shapes of distribution curves look alike. The correlation coefficient of link-hop and download rate is -0.09, indicating a weak negative correlation between them. Similarly, Fig. 7(b) shows the number of link-hops (left y axis) and the corresponding upload rates (right y axis) for BT probing connections in ascending order of link-hops in the TV swarm. The correlation coefficient between link-hop and upload rate is -0.19.

As shown in the above measurement results, we observe no strong correlation between the routing hops (link- or AS-hops) and the transmission rate (download or upload rate) for a connection, which implies that the download rate driven peer selection policy in native BT is not necessarily able to find a good peer with a high upload rate and small routing hops. For a BT peer in a swarm, there may exist multiple peers with similar download/upload rates but significantly different routing hops to this peer.

Therefore, for peer selection, we propose to use the ratio of download rate to upload rate, d/u, divided by link-level or AS-level routing hops, l or a, as a comprehensive metric. Using this metric, when peers have close d/u but their l or a vary greatly, the client can retain the download time while reducing the network traffic by selecting the peer with smaller hops. Note that we study the metric to select peers to unchoke, while seeds are not affected by the choke/unchoke mechanism. Thus, we only use this metric for leechers, whose upload rates u are always > 0. Fig. 9 shows the CDF of download-rate/upload-rate ratio divided by link-hops, $d/(lu)$, for different connections in the four BT swarms. As shown in the figure, about 1% - 5% connections have a $d/(lu)$ greater than 1. We have also studied the ratio of download rate to upload rate divided by AS-hops, $d/(au)$, for the same set of BT swarms, and find that 5% - 15% connections have a $d/(au)$ greater than 1. These results imply that our comprehensive metric can capture peers with high download rate, low reciprocal upload demand, and low routing hops.

C. TopBT Peer Selection Policy

In TopBT, there are several places to incorporate topology-awareness for better peer selection, namely, 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 similar principle can be applied to the other places as well.

TopBT Unchoking Mechanism

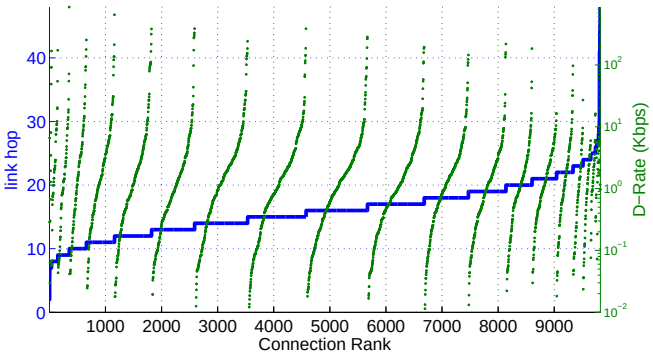
In essence, a TopBT client unchokes its connected peers round by round. The round duration is configurable, and each round usually lasts several minutes. At the end of each unchoking round, a TopBT client computes d, u, and l (or a), sorts connections based on metric $d/(lu)$ or $d/(au)$ in descending order, and picks connections with top metric values to fill its upload cap. Once the top connections are determined, the client unchokes them simultaneously. To reflect network condition dynamics, the TopBT client refreshes d, u, and l (or a) in each unchoking round.

At the starting phase, before there is data transmission, the TopBT client initializes d of every connection to its download capacity equal split, i.e., download capacity divided by the number of connections; and initializes u to the equal split rate that reciprocation is essentially assured as in BitTyrant [25], where Piatek et al. choose 14K Bps as the initial value of upload rate.

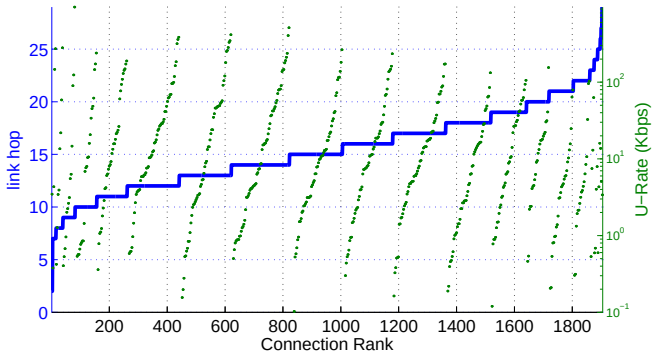
Once connections are established, in each unchoking round, the TopBT client can passively monitor d and u for mutually unchoked peers, and measure l and a using the link-hop examiner and the AS-hop examiner. However, for a connection choked on either side, TopBT yields d or u to 0. To handle this case, we use the estimation method in BitTyrant [25]. The client adjusts its estimated reciprocal upload rate for each peer based on whether that peer chokes it. If it is choked by that peer, it increases its estimated reciprocal upload rate for that peer by a constant factor; if it was unchoked by that peer consecutively in several past rounds, it reduces its estimated reciprocal upload rate by another different factor.

Handling Non-responding Peers

For those peers not responding to ping or traceroute probes, we use the average routing hops of responding peers as the estimation of their hops. Although it can mistakenly unchoke faraway peers with moderate download/upload ratios, 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,



(a) Link-hop and download rate.



(b) Link-hop and upload rate.

Fig. 7. Hop and Rate Relationship for TV.

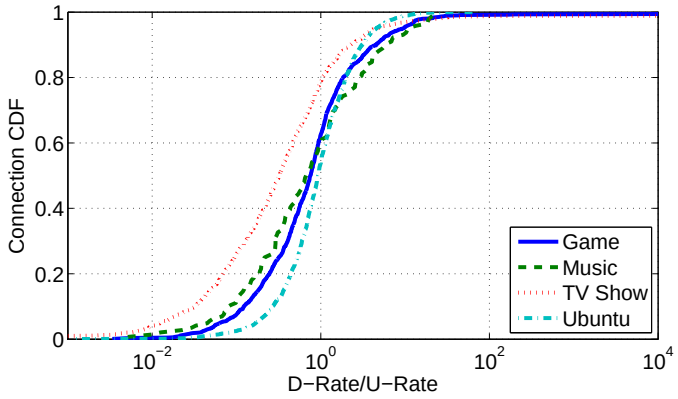


Fig. 8. Download rate divided by upload rate.

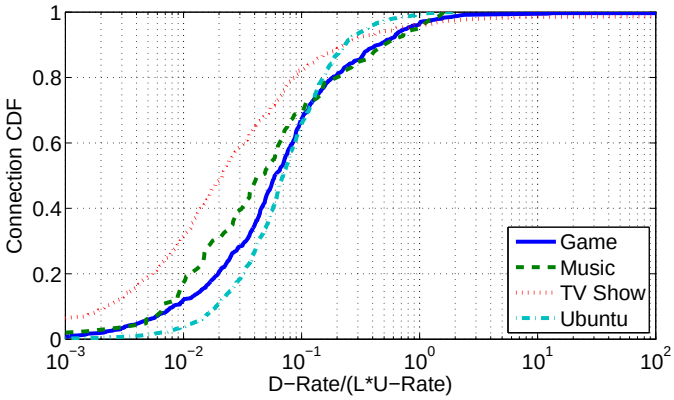


Fig. 9. Download rate divided by link-hops and upload rate.

it uses the download/upload ratio for sorting.

Problem 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. Incentive Mechanism For TopBT Deployment

ISPs can significantly benefit from wide deployment of TopBT clients in that both cross-ISP traffic and average link stress can be reduced. Therefore, they have strong motivation to encourage end users to use TopBT. However, from end users' perspectives, they care most about download time, i.e., how fast a file can be downloaded. To boost the incentive for end users to adopt TopBT, ISPs can prioritize TopBT traffic over the traffic generated by other BT clients, so that TopBT download time can be shorter.

V. Experiments and Evaluation

To evaluate TopBT's performance, we conduct large scale distributed experiments and compare against *native BT* and *BitTyrant* [25]. 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.

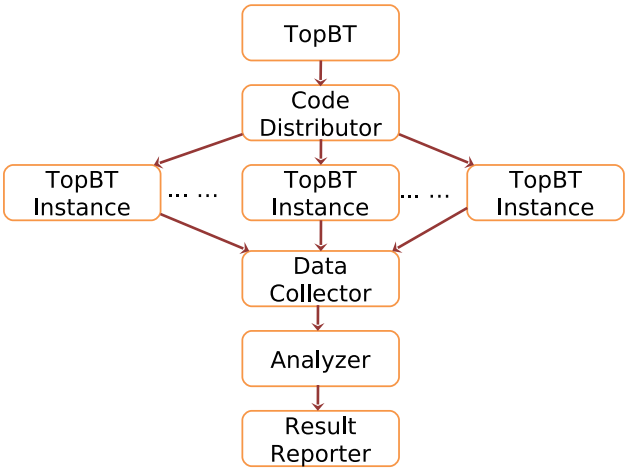
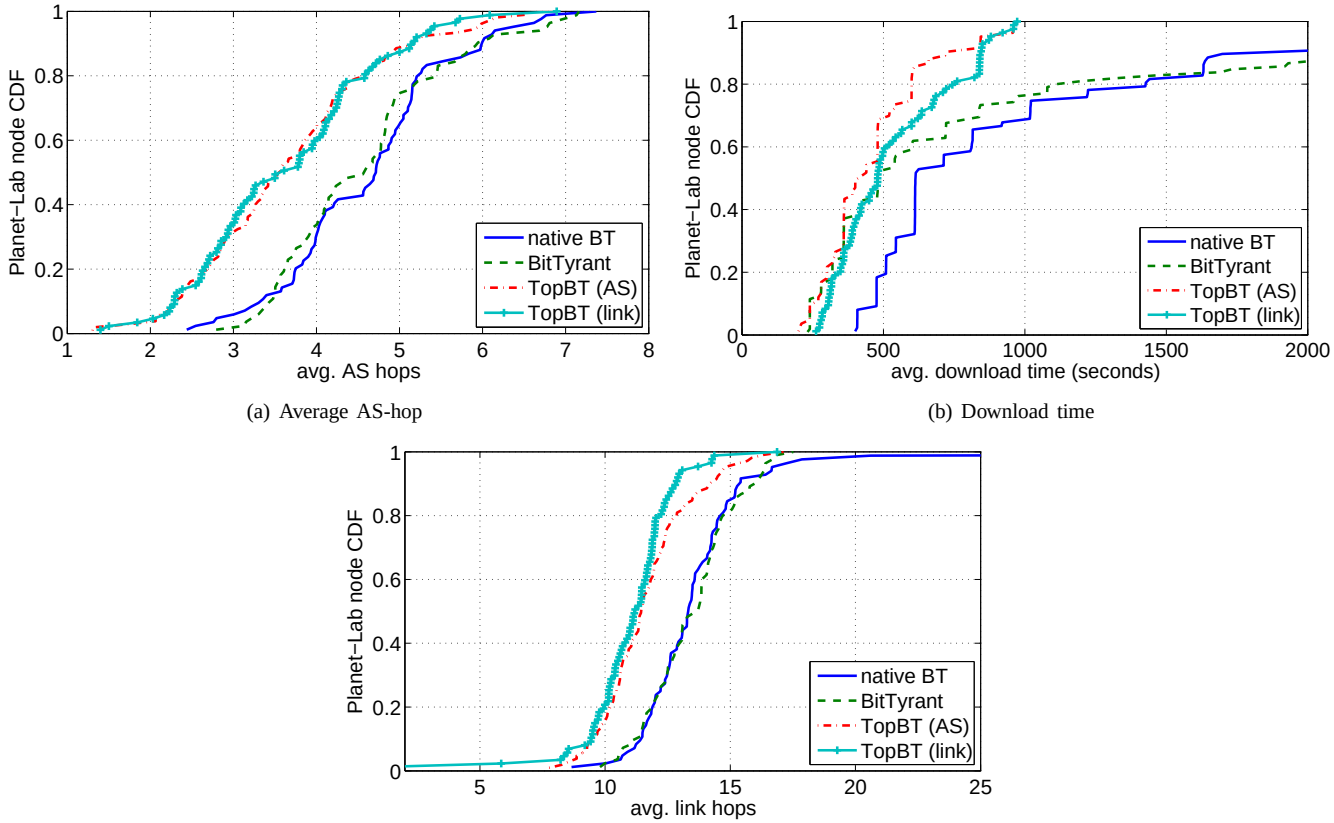


Fig. 10. TopBT deployment and experiment workflow.

We implement several tools to deploy TopBT to Internet hosts to conduct our experiments and collect data, as shown in Fig. 10. Specifically, the *code distributor* is run on our local machine, which replicates the TopBT code to a list of remote hosts, and instruments them to execute TopBT client program. After the program finishes on remote hosts, the *data collector*

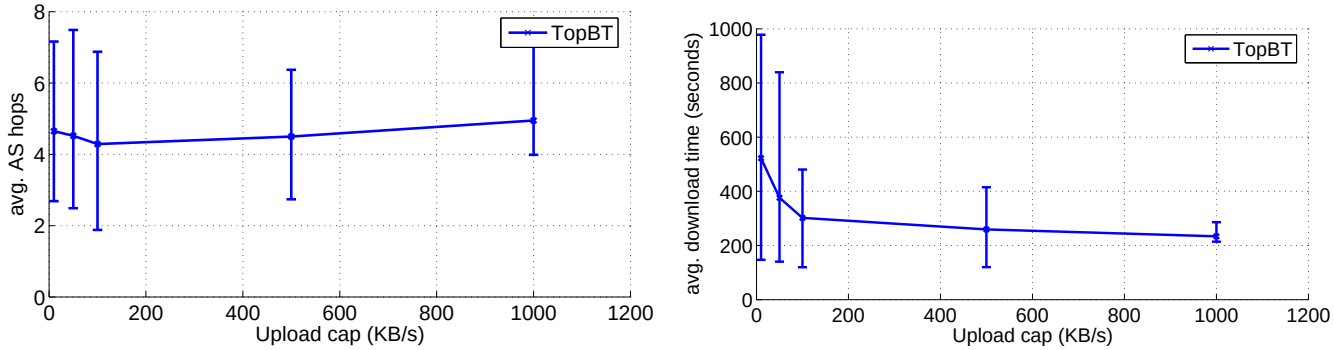


(a) Average AS-hop

(b) Download time

(c) Average link-hop

Fig. 11. TopBT experiment results.



(a) Average AS-hop

(b) Download time

Fig. 12. Real deployment results on PL nodes under upload capacity constraint (TopBT based on AS-hop).

is responsible for collecting the TopBT data from hundreds of remote hosts, and the *analyzer* aggregates all the data across hosts, and calculates and analyzes the performance from the aggregated data. Finally, the *result reporter* generates reports on performance metrics, such as download time and ISP traffic, etc.

We deploy 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 start native BT, BitTyrant, and TopBT in random order: when one completes, we randomly pick 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 are 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.

Although practically we cannot make the Internet situation the same when different clients were running, we have made every effort to minimize the impact of Internet dynamics. In addition, 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 also study 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 evaluate TopBT overhead in terms of the amount of probing traffic it triggers.

Traffic and Download Time

Fig. 11(a) shows that among the 100+ PL node experiments, link-hop and AS-hop based TopBT always achieves lower AS-hops. Compared to native BT and BitTyrant, 30% TopBT PL nodes have 25% less AS hops. Given the large size of the BT downloading file, it implies that TopBT can save a significant amount of Internet traffic. Link-hop based TopBT can achieve slightly lower AS-hops than AS-hop based TopBT, possibly because AS-hops cannot be obtained for a large fraction of peers due to that routers filtering traceroute packets, and the client cannot use the comprehensive metric to unchoke them.

For the download time, Fig. 11(b) illustrates that both TopBT and BitTyrant finish downloading about 15% faster than native BT, and TopBT's download time on all PL nodes is a little shorter than BitTyrant. AS-hop based TopBT can download slightly faster than link-hop based TopBT.

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. 11(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 AS-hop based TopBT, possibly due to its accurate measurement of routing hops.

Effects of Upload Capacity

Fig. 12(a) shows that the upload cap does not affect the average AS hops, while Fig. 12(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. 13 shows 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.

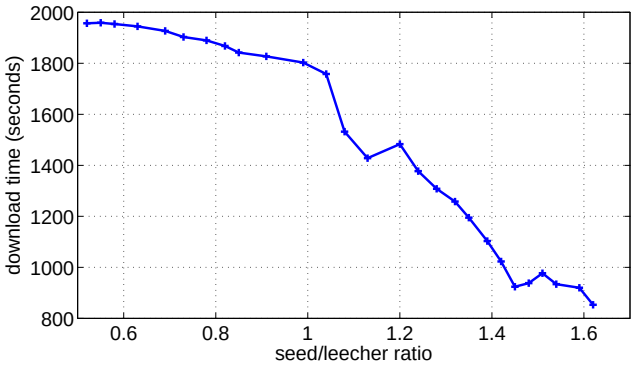


Fig. 13. Seed/leecher ratio effect on TopBT download time.

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 longer, about 40 seconds to complete probing. The probing process runs at 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. 14 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

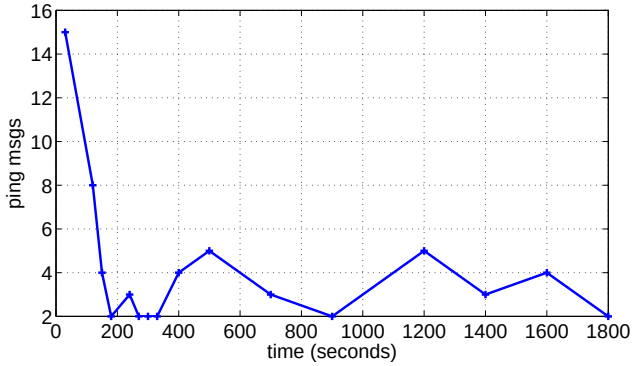


Fig. 14. TopBT ping messages generated during downloading.

messages generated.

Discussion

In the native BT protocol, the number of active download connections is limited because the upload cap can be reached easily. As a result, the number of active download connections is fewer than that in BitTyrant, and the aggregated total download rate is lower than that in BitTyrant and our TopBT protocols. For BitTyrant, in general, it performs well in download time. However, because it does not consider traffic saving on the Internet, it tends to waste a large amount of Internet traffic by downloading from more remote peers. The evaluation results show that TopBT can save a significant amount of Internet traffic while retaining the fast download speed for end users. We observe similar results from our residential host experiments, in which 14 hosts are geographically distributed in China, Europe, Canada, and U.S. with typical cable and DSL connection settings.

In order to gain more traffic saving, it is important that a TopBT host is aware of the large peer pool instead of only its currently connected peers, so that it can unchoke those closest peers with high download rate and low upload cost. TopBT clients always aggressively retrieve peer lists from the tracker site. By tuning the aggressiveness parameter, our TopBT client can effectively maintain a large list of peers. We have also done experiments by using a modified tracker site that can return almost complete peer list, in which the BT host constantly probes and measures its proximities to those peers. We observe large performance gain on the host in these experiments.

VI. Conclusion

The BitTorrent-based file downloading consumes the largest portion of Internet bandwidth now. However, the current BT applications have over-utilized 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. In this paper, we present TopBT, a topology-aware and infrastructure-independent BitTorrent client. We have designed and implemented TopBT based on open-source Vuze, BitTornado, and LH-ABC, and have extensively evaluated its performance. We show that TopBT can significantly decrease download time, and can save up to 25% induced Internet traffic, compared with several other representative BT clients. TopBT has been integrated into Vuze, a mainstream BT client, and has been downloaded for a significant number of times since its release. For more information about TopBT in practice, please refer to http://topbt.cse.ohio-state.edu.

References

[1] Internet Study 2007. http://www.ipoque.com/resources/internet-studies/internet-study-2007.
[2] Azureus. http://www.azureus.com.
[3] A. Bellissimo, P. Shenoy, and B. Levine. Exploring the Use of BitTorrent as the Basis for a Large Trace Repository. TR 04-41, UMass. Amherst, 2004.
[4] 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.
[5] BitComet. http://www.bitcomet.com.
[6] BitTornado. http://www.bittornado.com.
[7] D. R. Choffnes and F. E. Bustamante. Taming the Torrent. In *Proceedings of the 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM'08)*, Seattle, WA, USA, August 2008.
[8] Comcast and BitTorrent Form Collaboration. Associated Press, October 2007.
[9] C. Dale, J. Liu, J. Peters, and B. Li. Evolution and Enhancement of BitTorrent Network Topologies. In *Proceedings of the 16th International Workshop on Quality of Service (IWQoS'08)*, Enschede, Netherlands, June 2008.
[10] A. Fei, G. Pei, R. Liu, and L. Zhang. Measurements on delay and hop-count of the internet. In *Proceedings of 1998 IEEE Global Telecommunications Conference (IEEE GLOBECOM'98)*, Sydney, Australia, November 1998.
[11] Fileshareshark topbt review. http://filesharefreak.com/2009/01/23/topbt-on-vuze-a-topology-aware-bittorrent-client/.
[12] M. J. Freedman, K. Lakshminarayanan, and D. Maziéres. Oasis: Anycast for any service. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (USENIX NSDI'06)*, San Jose, CA, USA, May 2006.
[13] 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.
[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] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garc'es-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *Proceedings of the 2004 Passive and Active Measurement Conference (PAM'04)*, Antibes Juan-les-Pins, France, April 2004.
[16] 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.
[17] J. Ledlie, P. Gardner, and M. Seltzer. Network coordinates in the wild. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (USENIX NSDI'07)*, Cambridge, MA, USA, April 2007.
[18] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and Sharing Incentives in BitTorrent Systems. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS'07)*, San Diego, CA, USA, June 2007.
[19] LH-ABC: Enhanced Multi Platform BitTorrent Client. http://code.google.com/p/lh-abc/.
[20] 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.

[21] Z. Mao, J. Rexford, J. Wang, and R. Katz. Towards an Accurate AS-level Traceroute Tool. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM'03)*, Karlsruhe,Germany, August 2003.

[22] L. Massoulié and M. Vojnović. Coupon Replication Systems. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS'05)*, New York, NY, USA, June 2005.

[23] K. Parvez, C. Williamson, A. Mahanti, and N. Carlsson. Analysis of BitTorrent-like Protocols for On-Demand Stored Media Streaming. In *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS'08)*, Annapolis, MD, USA, June 2008.

[24] R. L. Pereira, T. M. S. F. V. Vasques, and R. M. 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.

[25] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (USENIX NSDI'07)*, Cambridge, MA, USA, April 2007.

[26] PlanetLab. http://www.planet-lab.org.

[27] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The BitTorrent P2P File-sharing System: Measurements and Analysis. In *Proceedings of the 4th Annual International Workshop on Peer-To-Peer Systems (IEEE IPTPS'05)*, Ithaca, NY, USA, February 2005.

[28] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM'04)*, Portland, OR, USA, September 2004.

[29] Default Initial TTL Values. http://members.cox.net/∼ndav1/self_published/.

[30] uTorrent. http://www.utorrent.com.

[31] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: a lightweight network location service without virtual coordinates. In *Proceedings of the 2005 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM'05)*, Philadelphia, Pennsylvania, USA, 2005.

[32] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Provider Portal for Applications. In *Proceedings of the 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM'08)*, Seattle, WA, USA, August 2008.

[33] X. Yang and G. Veciana. Service Capacity of Peer to Peer Networks. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'04)*, Hong Kong, China, March 2004.