Torrent Crawler: a tool for collecting information from BitTorrent networks

Yeounoh Chung

1. Abstract

BitTorrent is a free peer-to-peer (P2P) content-sharing application with a complex and dynamic overlay structure due to loose coupling, high churn rate, and varying responsiveness of nodes. The complexity and the dynamic nature of the overlay structure can mask the problems in the network, making errors difficult to detect and diagnosis in a timely manner. Furthermore, the heavy reliance of clients on the node local views compounds the problems such as partitioning in the network or load imbalance due to biased peer selection.

In an effort to provide the network with partial global information to resolve the network problems, this project looks into introducing a tool that efficiently collects global information from BitTorrent network. The tool, called Torrent Crawler (TC) uses a number of techniques to efficiently find all participating peers of the swarm, collecting global information from the network. The crawler also collects the information unobtrusively to the network traffic. In this paper, we describe the design, the implementation, and the evaluation of TC.

2. Introduction

BitTorrent[8] is a free P2P content-sharing application that enables a peer to distribute its content to a large number of peers without having a large upload bandwidth. A peer can utilize the aggregated upload bandwidth of the torrent swarm by duplicating its content over several peers; a peer, who wishes to download the content, downloads different parts of the content from different peers simultaneously. To download a file, a peer will first get a torrent file of the content published by a seeder, a peer with the entire file. The torrent file contains the URL of the centralized coordinator, called tracker, of the torrent network, and the file's metadata information hash. The downloading peer then contacts the tracker to discover the peers to download the file from.

BitTorrent has gained a lot of popularity in recent years. The loose coupling of nodes and the structural redundancy to handle high churn rates and varying node responsiveness of BitTorrent network results in a much complicated behavior of the network. The complexity of the overlay network structure and the unpredictability of the clients can cause problems in the network, making errors difficult to detect and diagnosis. The reliance of a node on node local views to maintain the overlay structure further compounds the problem. Defects or anomalies such as partitioning in the overlay or load imbalance due to biased peer selection are undetectable without partial global information and a good understanding of the various characteristics and dynamic behaviors of the network. However, such an understanding was often times missing or incorrectly obtained from non-representative measurement studies, in which a few instrumented clients were used to capture the desired properties.

Collecting representative global information such as peer's download rate, known neighboring peers, and the network's churn rate, from the complex and dynamic BitTorrent network is not simple. One way is to run a customized tracker to track the distribution of particular content by publishing a torrent file. BitTorrent tracker will work a centralized coordinator for the torrent file, making every downloading client to periodically contact the tracker and sending the list of peers to each client for its download. However, using a customized tracker to collect global information has some drawbacks [1]: First, each peer updates tracker of its download and upload information once every 30 minutes. This implies that variations on measurement in shorter time scale are not available. Second, the tracker does not know accurate information about connectivity of individual peers. The tracker update message from a client does not contain the client's connectivity information; a client could use a gossip like protocol to exchange its known peer list with other clients. Third, the tracker log does not provide any information about peers' maximum achievable download and upload rates.

Another approach to collecting information from the torrent network is to use several instrumented clients to capture their observed performances [4][5]. An instrumented client is a normal BitTorrent client that can perform measurement on its observed network status and received message statistics. But, this approach does not provide a representative view of all the participating peers or group information such as the

number of seeders in the network, the availability of pieces [1].

Instead of the above two approaches, TC uses the network crawling based approach to collect global information, interactively communicating with both the tracker and the peers of any given torrent. In order to speed up the crawling, first, TC requests the tracker of more peers frequently than normal BitTorrent clients. Second, TC uses BitTorrent Peer Exchange Protocol (PEX) to discover peers known to others from the other peers. And finally, TC advertises itself as a seeder, a peer with whole content, to the tracker and peers to have undiscovered peers to connect to TC. Using TC to collect global information on all the participating peers and the torrent network, we plan on collecting a representative view of all the participating peers in about 8 minutes.

Although, BitTorrent relies on client altruism for file sharing, because TC advertises itself as a seeder and does not upload any contents, other clients might punish TC for such behavior [9][10]. Fortunately, the punishment takes a form of reducing the bandwidth allowed to TC for downloading from the client, which is irrelevant to TC's operation. BitTorrent community also maintains a list of IP ranges to block peers from unwanted organizations. This IP blacklisting is to prevent certain organizations from accessing the network and is not an issue for TC.

Another interesting aspect of BitTorrent studies is BitTorrent's contribution to network traffic. In this work, we make the measurement unobtrusive to the network. To achieve this goal, TC never actually downloads, requests, or uploads files to observe network performance; it only uses the BitTorrent messages sent by other peers and the tracker to measure performance and piece availability. TC only sends out a single handshake message to each peer and the tracker at the beginning of the connection (or reconnection). Although, TC contacts the tracker more often than it is advised to get the global population, it would not burden the entire network much.

In addition, Torrent Crawler uses Amazon Simple Storage Service (S3) to store the collected information. Amazon S3 is highly scalable, reliable, and available distributed storage system with a simple and extensive API library. Amazon S3 provides a simple, reliable solution for any number of clients to view the stored results. Furthermore, Using Amazon S3 to store and to publish the collected information separates this data storing and organization layer from the data collection layer of the system, without putting much extra implementation efforts. Once the collected information is stored in S3 as public accessible objects, clients can use web browser to view the information.

In this survey, we present the design, the implementation, and the evaluation of Torrent Crawler, a tool to efficiently collect global information. By interactively communicating with both the tracker and the participating peers of the given torrent network, TC can provide a representative view of the swarm promptly. TC relies on BitTorrent messages sent by other peers to collect information, instead of exchanging any actual contents to observe network performance, making the measurement unobtrusive to the network traffic. Finally, the collected information is stored in Amazon S3 for reliable accesses from any number of clients, and to simplify data storing and organization.

3. Related Work

To gain a good understanding of various characteristics and dynamic behaviors of BitTorrent network, a lot of studies have been conducted from statistical modeling to simulation, and measurement. And to gain the good understanding that is also representative of the entire network, people have tried using a customized BitTorrent tracker for a given torrent file to monitor global information. However, BitTorrent tracker can only passively collect information from each peer once every 30 minutes. However, this approach has limitations in measurement time scale and efficiency. Many dynamics and variations happen in much shorter time scale will be lost. More importantly, the approach cannot detect the individual peer connectivity to detect network topology, biased peer selection, and load imbalance [1]. Hence, it is desirable to have a more efficient and reliable tool for collecting global information of the network.

A more common approach to study BitTorrent network is to use instrumented clients to observe their performances in the torrent swarm [4][5]. While instrumented clients provide accurate information and measurement of the network performance from the clients' standpoints, the collected data often fails to represent the entire population of the torrent swarm [1].

In studying the effect of the torrent attacks initiated by media corporate and movie production studio, a group of people have used a crawling based technique to discover all the participating peers of a given torrent file [2]. They claim that their crawler is able to discover most of the participating peers (i.e. over 90% of the entire population) just under 8 minutes, and their crawler contacts the tracker as well as using Azereus's gossip based protocol to discover

more peers. TC also contacts the tracker to get more peers, but more frequently; TC uses PEX instead of Azereus's gossip protocol to discover peers from other peers in the swarm. We believe using PEX instead of Azereus's gossip protocol is better, because a more BitTorrent clients including the most popular µTorrent support PEX. In addition, TC accepts incoming connections from undiscovered peers from the swarm.

4. System Design

Torrent Crawler crawls peers in a given torrent swarm and performs measurement on each peer using its received messages, without sending any unexpected or disruptive messages. The crawler is designed to efficiently collect global information of the network. This section describes the overall architecture of the crawler system and some important design choices made.

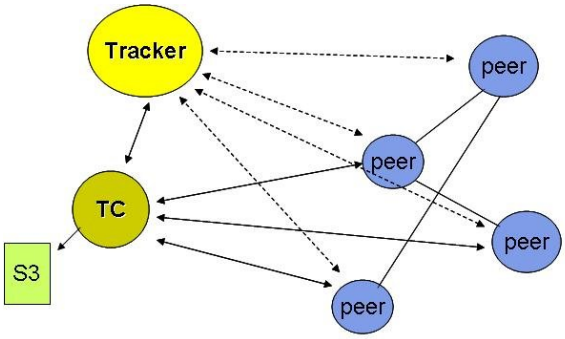

Figure 1 System Overview. Torrent Crawler (TC) contacts a tracker to discover a set of new peers; it connects to the peers. TC also accepts incoming connections from other peers. At the end of its operation, TC stores all its collected information to Amazon S3

4.1 Architecture

Torrent Crawler is a tool to efficiently collect global information given a torrent file. In order to collect representative global information of the torrent network efficiently without using either a customized tracker or an instrumented client, TC needs to crawl the vast majority of peers in the given torrent swarm in a timely manner. To achieve this goal, TC uses a number of techniques to speed up the crawling process: First, it advertises itself as a seeder to attract other undiscovered peers. Another benefit is that TC can connect to the peers behind Network Address Translation (NAT) box by accepting the incoming connections from the peers. Second, TC requests a tracker of peer addresses more often than it is advised to by the tracker. The interval between consecutive requests is chosen to be 2 minutes to request frequently enough without getting blocked often by the tracker. Finally, we also plan on using Peer Exchange Protocol which allows peers to exchange the information needed to find and connect to peers [7]. PEX protocol is used in one of the most popular BitTorrent client, µTorrent as well as in other clients including Vuzu (formerly called Azereus).

Once Torrent Crawler discovers a connectable peer, it stores the information of the peer (e.g. pieces downloaded, latency, download rate, etc.) and listens to their socket channels for any incoming messages. The received messages will be used to perform any peer-level and group-level measurements.

After TC finishes crawling the network, it stores the collected information as a public accessible, and web-viewable object in Amazon S3 server.

4.2 Communication with Tracker

TC communicates with a tracker via HTTP channel, given an URL of the tracker from a torrent file. Because Domain Name Server (DNS) lookups are blocking, the connection between TC and a tracker is intrinsically synchronous; depending on the tracker's states and network traffic, TC's blocking requests to the tracker may take up to a few seconds. And since TC requests a tracker more often than it is supposed to, the tracker may stop listening to TC for a period of time.

In order to continuously listen to peers without stopping for periodical blocking requests to tracker, we separate the asynchronous connections to peers with the synchronous connection to tracker. TC spawns a separate thread for handling a connection with a tracker; the thread dedicated to the connection uses message piping to deliver peer lists received from the tracker to the main thread, without interrupting the main thread.

4.3 Communication with Peer

Because of the asynchrony of the BitTorrent Messages other than 'handshake' message, TC uses non-blocking channels to communicate with peers. TC can either connect to or accept a connection from a peer. When TC receives an IP address and a port number of an unknown peer from the tracker, TC initiates a connection to the unknown peer using a non-blocking channel. And TC also advertises a local port bound to a non-blocking socket to the tracker for accepting connections from any unknown peers. Once TC establishes a connection with peers, TC listens to the channels for any incoming BitTorrent messages. Finally, TC does not send periodical 'keep

alive' messages to peers to prevent peers from closing the connections after not receiving any messages from TC. If a connection between TC and a peer closes, then TC reconnects to the peer after a time-out. This allows TC to reduce the number of BitTorrent messages in the network and to take several latency measurements on the particular peer, without aggressively reconnecting if the peer intentionally closed the connection.

4.4 Measurement

TC keeps track of the states of the discovered peers of the given torrent. The states includes IP address, port number, latency, download rate, connection status, available pieces of the content, PEX support status. To measure latency of outgoing connections, TC initiates a non-blocking connection to each peer and times until the connection is established; we use WireShark to monitor Transmission Control Protocol (TCP) packets on the advertised port to measure incoming connection latency. In BitTorrent, each peer sends out a 'have' message for each newly downloaded piece of the content. Because peers do not send 'have' messages right after they finish downloading, TC estimates the download rate of each peer by counting the number of 'have' messages received from the peer over some time period. Finally, TC aggregates the information from peers and the tracker to obtain the torrent network level information, such as content availability by piece, the total number of seeders/leeches known to the tracker, etc. We plan on getting the measurement done by sending just BitTorrent 'handshake' messages to establish connections to minimize TC's effect on the network traffic.

4.5 Storing Results in Amazon Simple Storage Service (S3)

Amazon S3 provides a simple web-storage interface to store and retrieve from web. The storage system is highly scalable, reliable, and available. Using Amazon S3 simplifies storing and organizing the results of different torrents with different sizes. Amazon S3 lets TC to store data as an object in a bucket with a unique key; TC can simply read and write to the objects via a simple interface. Lastly, TC can store any number of objects of any sizes in S3, without concerning over the scalability of the storage system.

5. Implementation

Torrent Crawler uses a blocking HTTP protocol to communicate with a tracker and non-blocking Transmission Control Protocol (TCP) channels to communicate with peers. The synchronous connection between TC and a tracker is handled by a separate thread to avoid introducing blocking delays while continuously listening to the asynchronous connections with peers. Each time TC requests the tracker of more peers by sending a tracker 'handshake' message, TC advertises itself to the tracker as a seeder with a port number it is listening on (between 6881 and 6889) to attract other peers for incoming connections; the tracker sends back a list of IP addresses and port numbers of a number of peers. Similarly, when the other peers on the network requests the tracker, they will receive the IP address and the port number of TC.

Once TC learns about a new peer, TC registers the peer to its selector for a sequence of operations. First, TC initiates a connection and starts measuring latency to the peer. In order to keep the number of registered peers reasonable, TC will deregister any unreachable peers after timeouts. Second, after successfully connecting to the peer, TC exchanges 'handshake' messages and receives a 'bitfield' message from the peer. A 'bitfield' message contains information about the sender's possession of the file content. Lastly, upon receiving the 'bitfield' message, TC starts estimating download rate of the peer by counting the number of 'have' messages sent by the peer over a period of time. A 'have' message is sent for each newly downloaded piece by the sender. Once TC performs all three operations, TC deregisters the peer from its selector, keeping the number of registered peers reasonable over a long period of time.

Whenever the selector times out or idles with no registered peers, TC registers a new peer from a queue to the selector; TC requests the tracker of more peers if there is no more peer to register. This policy for requesting the tracker makes TC requests the tracker quite often.

In its 'handshake' response, the tracker also specifies the interval a client should wait before re-requesting the tracker, which is generally much longer than 2 minutes; TC requests much aggressively than it is advised to until some threshold number (~90%) of discovered peer is reached given the total number of the peers in the swarm from the tracker. Note that the tracker is a centralized coordinator that every peer in a torrent swarm must contact to at least once. Because of this abnormal behavior of TC, the tracker might block TC from contacting the tracker for a period of time, in which case, TC simply retries later. The interval TC waits before retrying varies, depending on the number of

peers TC is currently monitoring and the number of new peers that TC knows about but has not connected to, yet.

As mentioned previously, connections between TC and peers are asynchronous and use TCP; TC continuously listens to the TCP sockets associated with the peer connections. Because messages from a peer arrive as a stream of bytes at each socket, TC often receives fragmented messages. Defragmentation of messages is simple because TCP guarantees an in-order delivery of message bytes and each BitTorrent peer message is prefixed by a fixed size header with a message body length filed: TC buffers a byte stream on each socket until a complete message is received.

After a user specified crawling timer expires, the crawler stores the message log of every messages received during the crawling period, the history of network connectivity and torrent file availability over time, and the states of every discovered peer on Amazon S3.

6. Evaluation

We have implemented a working prototype of TC without PEX protocol. The prototype was installed on a Linux server machine (schroeder.csuglab.cornell.edu) for testing; we have used actual torrent files, without having TC actually download any of the distributed content.

In this section, we describe the results of running Torrent Crawler with six different movie Torrent files. The crawling time for each Torrent was 10 minutes.

6.1 Connectivity

The main goal of TC is efficiently collecting the global information of a given Torrent network. To achieve this goal, TC aggressively communicates with the tracker and accepts any incoming connections. Unfortunately, we have seen one or two incoming connections per single ten-minute crawling. Because the tracker randomly chooses a set of peers to fetch to each requesting peer, there are very few incoming connections. Furthermore, we suspect that the firewall on the server machine could have affected incoming connections. In the future, we plan on isolating the effect of the random peer selection of the tracker by disabling the firewall.

The random selection of peer set by the tracker can also limit the rate TC discovers new peers once TC discovers a large portion of the entire peer population.

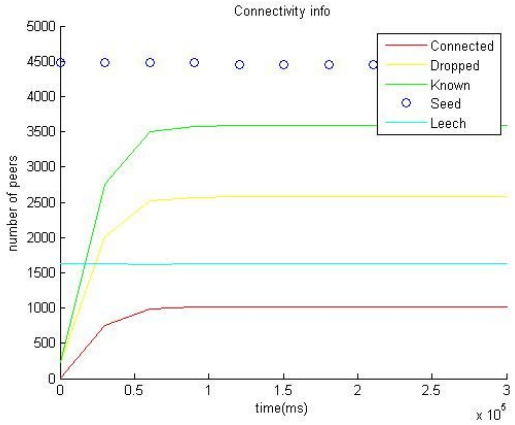

Figure 2 Connectivity information of the Torrent network associated with 'Twilight.2008.DvDrip-NoRar__.4752496.TPB'

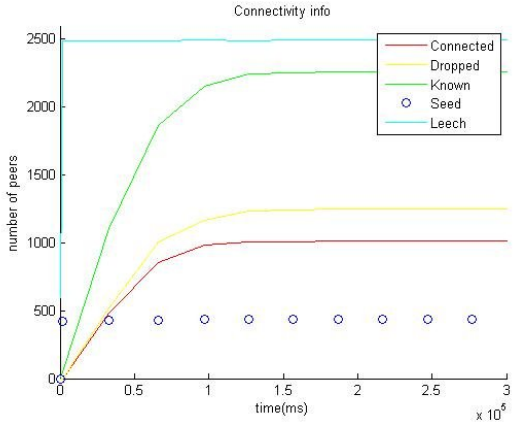

Figure 3 Connectivity information of the Torrent network associated with 'Flirting+with+40+1337x-X.avi'

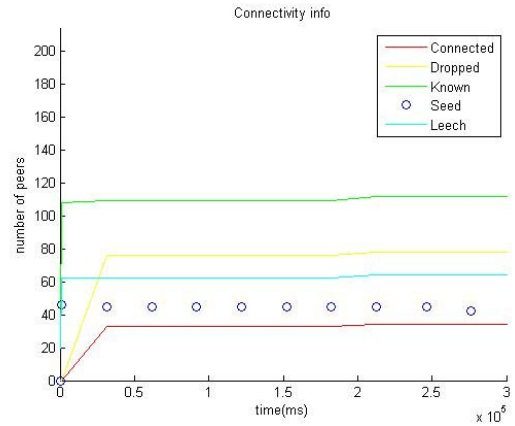

Figure 4 Connectivity information of the Torrent network associated with 'Kismat_Konnection_2008_DVDRip_XviD_SaM-++Demonoid.com++'

Figure 2 shows that TC discovers some portion of the entire participants (seeders and lechers combined) of the network associated with Twilight movie Torrent file and Flirting in about a minute; after a minute, the number of 'known' or discovered peers stays almost constant because tracker responses then contain peers that are already known to TC. On the other hand, TC discovers almost all of the peer population in Figure 4, where there are only a few hundreds peers in the entire swarm.

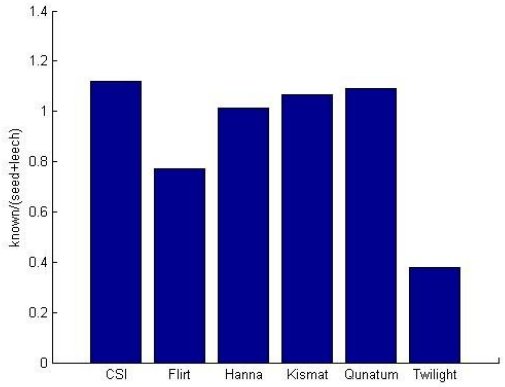

Figure 5 Known (i.e. discovered) peers to the size of swarm ratio at the end of 10 minutes crawling for each Torrent network. Some of the ratios are bigger than one because known peers include peers that have disconnected from the networks.

The results demonstrate that TC discovers most of the entire swarm, if a Torrent network is relatively small (i.e. contains a few hundreds peers). When a Torrent network contains thousands of nodes, then the effect of the random peer selection of the tracker dominates, and TC rarely receives a new peer from the tracker after some time.

6.2 Download Rate

BitTorrent provides good bandwidth utilization by allowing its users to perform bilateral exchange of blocks; each user can utilize more bandwidth by downloading different blocks from different neighbors concurrently.

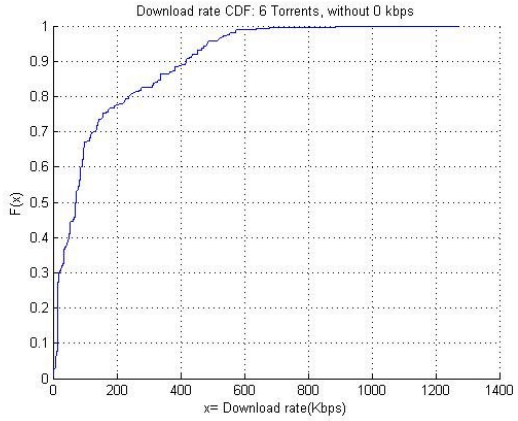

Figure 2 CDF of download rate of leechers (for all six Torrents).

Figure 6 demonstrates that the majority of leeching peers over all six Torrents experience aggregate download rate of 100 kbps, which is likely because there are more peers in the middle of their downloads. A typical BitTorrent client experiences slower aggregate download rates at the beginning and at the end of downloads, ranging from 10 kbps to 300 kbps, when they have fewer blocks to trade for new blocks or have a few blocks to download, respectively.

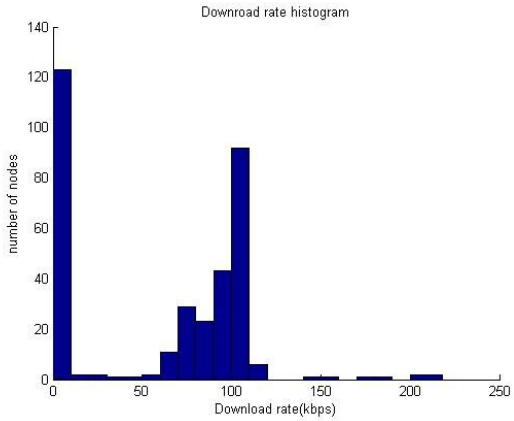

Figure 3 Aggregate Download rate of peers of the Torrent network associated with 'Twilight.2008.DvDrip-NoRar__.4752496.TPB' The peak at 0 kbps is due to a large number of seeders within the network.

Figure 7 shows the download rate distribution for a single Torrent file. The peak at 0 kbps is likely because there are a lot of seeders, who do not download any file content. And the download rates for leechers are concentrated around 100 kbps, which implies that there are more leeching peers in the middle of their downloads. Interestingly, the distribution (neglecting the seeders) is skewed to the

left, with a sharp cliff near 110 kbps. This is likely because download or upload bandwidths of most of the peers within the network are less than 100 kbps. Having small upload bandwidths limits download rates of users because most BitTorrent client applications limit a user's download rate based on the upload rate. In general, ISPs provide each user with higher download bandwidths and lower upload bandwidths.

6.3 Latency

TC observes its connection links to other peers to infer link latency to each peer. The link latency values between TC and peers within six different Torrent networks range from 0 to 3000 ms, which should be a rare occurrence given a typical TCP timeout value is 1500 ms for the first timeout and 3000 ms for the second timeout.

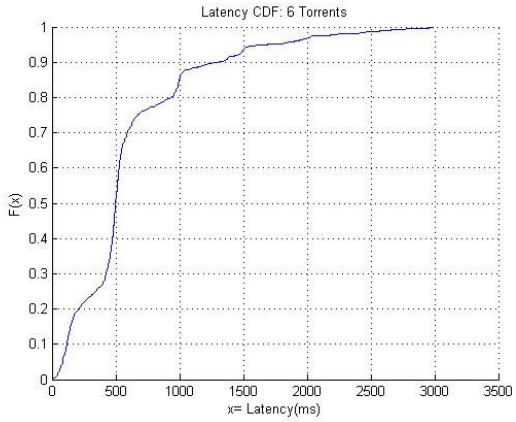

Figure 4 CDF of link latencies between TC and peers (for all six Torrents)

Figure 8 shows that less than 6 percents of the connections over the six Torrent networks experience link latencies bigger than 1500 ms. Furthermore, a steep jump at 500 ms implies that there are two modes in the distribution. About 50 percents of the link latencies are less than 500 ms, and another 30 percents of the link latencies are between 500 ms and 1000 ms. The steep jump is likely due to the inter-continental propagation delay; the two modes could represent a group of peers in North America and a group of peers in other continents, respectively.

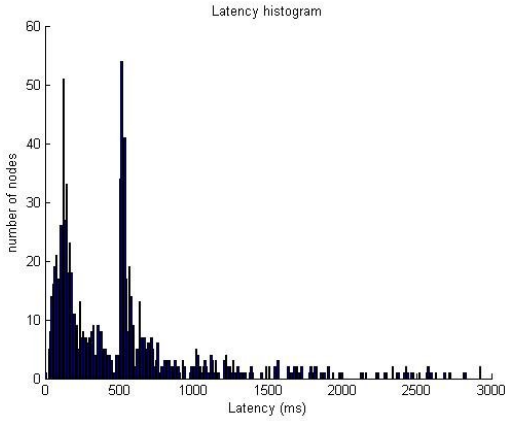

Figure 5 Latencies of connections between TC and peers of the Torrent network associated with 'Twilight.2008.DvDrip-NoRar__.4752496.TPB'

Figure 9 shows the observed latency distribution of the Torrent network associated with Twilight movie Torrent file. As observed in the aggregate CDF latency distribution, there are two modes; the majority of link latencies are less than 500 ms. This is likely because more BitTorrent users in the United States are sharing the Hollywood movie 'Twilight.'

6.4 Availability

Each BitTorrent client uses the rarest first algorithm to download relatively rare available pieces of content first. As a result, rare pieces of Torrent networks will become more available. Even though, the client relies on its local view to infer rareness of each block, our results demonstrates that the rarest first algorithm works well.

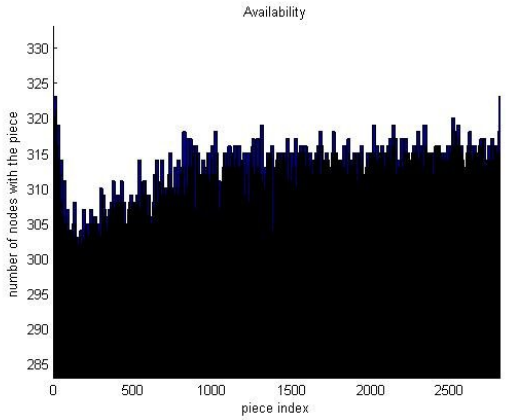

Figure 6 Availability of pieces of 'Twilight' movie content after 10 minutes of crawling. This shows no single rare piece, which could be the bottleneck for a peer's download completion.

7. Future Work

The prototype currently works with single file Torrents and peers with IPv4 addresses. Although, the majority Torrents available online are single file and contain peers with IPv4 addresses, we plan on extending TC to use multi files Torrents and peers with IPv6 addresses.

Another important future extension to TC is the use of PEX protocol. In order to speed up the crawling process, TC needs to implement PEX protocol. If TC connects to a peer with PEX protocol support, then TC sends a 'PEX handshake' message and receives the peer list of the connected peer. Also, TC can form a network topology of a given Torrent network, with the individual connectivity information from PEX protocol.

Lastly, we plan on having TC to use a coordinate-based mechanism proposed by S. Eugene Ng to measure inter nodes latency [11].

8. Conclusion

This paper describes the design, the implementation, and the evaluation of TC, a tool for efficiently and unobtrusively collecting global information from Torrent networks. TC efficiently collects global information from Torrent networks, by interactively communicating with both a tracker and peers; TC only uses a few received BitTorrent messages to measure important peer-level information, such as latency, download rate, and possession of each peer, without downloading any file content.

The evaluation results of TC show that TC can collect partial global information of Torrent networks under a few minutes. We will continue to improve the TC system to ultimately provide a centralized system with sufficient network global information to monitor and manage BitTorrent networks.

9. References

[1] A. Rasti, R. Rejaie Understanding Peer-level Performance in BitTorrent: A Measurement Study. In IEEE 2007

[2] P. Dhungel, D. Wu, et al. A Measurement Study of Attacks on BitTorrent Leechers.

[4] M. Izal, G. Urvoy-Keller, E. W. Biersack, *et al.* Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM*, 2004.

[5] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest First and Choke Algorithms Are Enough. In *IMC*, 2006

[6] T. Isdal, M. Piatek, et al. Leveraging BitTorrent for End Host Measurements.

[7] "http://en.wikipedia.org/wiki/Peer_exchange" last visited on April 22th

[8] Bram Cohen. BitTorrent Specification. "http://www.bittorrent.org/beps/bep_0003.html" last visited on April 26th

[9] "http://www.codinghorror.com/blog/archives/000795.html" last visited on April 26th

[10] "http://cubist.cs.washington.edu/Security/2009/03/13/security-review-bittorrent/" last visited on April 26th

[11] . S. Eugene Ng and Hui Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches", *IEEE INFOCOM'02*, New York, NY, June 2002