Understanding Peer Exchange in BitTorrent Systems

Di Wu[†], Prithula Dhungel[§], Xiaojun Hei[‡], Chao Zhang[§], Keith W. Ross[§]
† Sun Yat-Sen University, Guangzhou, China
‡ Huazhong University of Science and Technology, Wuhan, China
§ Polytechnic Institute of NYU, Brooklyn, NY, USA

***Abstract*—Peer Exchange (PEX), in which peers directly exchange with each other lists of active peers in the torrent, has been widely implemented in modern BitTorrent clients for decentralized peer discovery. However, there is little knowledge about the behavior of PEX in operational systems. In this paper, we perform both passive measurements and Planetlab experiments to study the impact and properties of BitTorrent PEX. We first study the impact of PEX on the download efficiency of BitTorrent. We observe that PEX can significantly reduce the download time for some torrents. We then analyze the freshness, redundancy and spread speed of PEX messages. Finally, we also conduct large-scale Planetlab experiments to understand the impact of PEX on the overlay properties of BitTorrent.**

***Index Terms*—BitTorrent, Peer Exchange, Measurement**

I. Introduction

Today, BitTorrent is one of the most popular P2P file distribution protocols, particularly for the distribution of large files such as movies, television series, record albums, and open-source software distributions. According to [1], the number of downloads of .torrent files from a leading torrent-discovery site *MiniNova* [2] reached about 7 billion in 2008.

To enable the trading of file chunks, it is essential for a peer to discover other peers in the same torrent. A common approach for peer discovery is to use a centralized tracker: each peer in the torrent registers with the tracker, and any peer can contact the tracker at any time to obtain a random subset of other peers in the torrent. Modern BitTorrent clients (e.g., uTorrent, Azureus, BitComet) additionally provide decentralized peer discovery using *Distributed Hash Tables* (DHTs) and *Peer Exchange* (PEX). A DHT provides the same tracking service as that of the central tracker in a distributed manner. Peers can query the DHT interface to obtain peer lists.

Different from DHT, PEX allows peers in a torrent to exchange lists of active peers directly with each other. PEX has been first implemented in Azureus to reduce the load on trackers and later also adopted by other popular clients (e.g., uTorrent, Mainline, etc). After initial bootstrapping, peers then only depend on other peers implementing the same PEX protocol to discover new peers. Currently, there is no official standard of PEX protocol. Multiple versions of PEX protocols have been developed and implemented by different clients.

Although PEX is widely deployed in BitTorrent systems, there lacks a comprehensive study of PEX in operational systems. In this paper, we perform both passive measurements and Planetlab experiments to study the impact and properties of BitTorrent PEX. We focus our study on the most popular PEX protocol, UT_PEX. Our main contributions and some important observations are listed as below:

- Among the tested torrents, we observe that about 70% of peers support UT_PEX, 15-20% of peers support AZ_PEX and only about 5% of peers support BC_PEX. Peers supporting those three PEX protocols account for 95% of all the peers.
- Using instrumented clients, we collect detailed traces of PEX activity as well as connection status. We observe that PEX enables the client to quickly acquire a large number of peers in the torrent and create more outgoing connections. In our tested 100 torrents, we find that PEX can increase the download speed of 40% torrents. The average reduction of download time is about 7%.
- We design a set of experiments to evaluate the freshness, redundancy and spread of PEX messages. We find that: (1) about 30% of PEX messages are completely fresh and over 80% of PEX messages have a freshness ratio greater than 0.5; (2) there exists significant redundancy in the PEX messages, and 20-40% of peers were repeatedly observed over 5 times; (3) the spread of peer information is not as fast as expected. It is difficult for peers to learn the complete peer list in the torrent within a short period.
- To gain a deep understanding of PEX in BitTorrent, we perform large-scale Planetlab experiments, in which we set up our own tracker, seed and leechers. We observe that PEX can enrich the connectivity of the whole BitTorrent overlay to a certain degree. In our experiment, when PEX is used, we do not observe the chain-like topology as reported in [3].

This paper is structured as follows. Section II provides the background information about main peer discovery mechanisms. Section III presents the detailed results obtained from passive measurement and Planetlab experiments. Section IV describes the related measurement work on BitTorrent. We summarize our work in Section V.

II. Background

One major component of BitTorrent is the peer-discovery component wherein a peer desiring to download a particular file using a BitTorrent client, has to discover other peers that are currently sharing the same file. This group of peers in the same BitTorrent download are referred to as the *torrent* for the file. Once a number of peers in the torrent are discovered, a peer then establishes TCP connections with these peers and starts trading for pieces of the file.

The conventional approach to discovering peers in a torrent consists of requesting the list of peers in the torrent from a centralized entity known as the *tracker*. This centralized approach to peer discovery is not scalable, especially for trackers that keep track of millions of peers for thousands of files. The problem of scalability has already been observed for some popular trackers such as Pirate Bay trackers, which often require peers to send multiple requests due to TCP time-outs.

In order to alleviate the load on trackers, two decentralized approaches to peer-discovery are being used by a number of clients: *Distributed Hash Tables* (DHTs) and *Peer Exchange* (PEX). A DHT is a database distributed over a network of computers, referred to as nodes. All nodes can store (key, value)-pairs on the DHT and later find values by searching for keys. Store and search actions are performed by finding the node responsible for the search key and then sending it a STORE or FIND_VALUE command. Note that, in order to join a DHT, a peer should first know a bootstrapping node in that DHT. Currently, the BitTorrent ecosystem consists of two major DHTs: the Azureus DHT and the Mainline DHT. The Azureus DHT is used only by Azureus clients and the Mainline DHT is used by all the other clients that support the DHT feature, including uTorrent, BitComet, and Mainline.

In the second decentralized method of peer discovery - PEX - peers *gossip* with each other in regular intervals with lists of active peers they know. However, this form of peer discovery first requires a peer to know at least one other peer using some other form of peer-discovery (e.g., using the centralized tracker). Furthermore, like DHTs, peers support exchanging peer lists only with other peers that use the same PEX protocol. Since there is no official specification for the peer exchange protocol, three major implementations of PEX exist in the BitTorrent world as of now: AZ_PEX [4] used by Azureus clients; UT_PEX [5] used by uTorrent, KTorrent, Opera, qTorrent, libtorrent, Deluge, MooPolice, Transmission, Aria clients; and BC_PEX used by BitComet clients. For more details about the PEX support of different clients, please refer to the article [6].

III. MEASUREMENT

In our measurements, we first developed a Java-based peer crawler to determine the distribution of BitTorrent clients that support different versions of PEX protocol.

Our peer crawler collects peer lists from the tracker and contacts each peer using the BitTorrent protocol. The successfully connected peers provide a random sample of the whole torrent. From the results shown in Table I, we observe that about 70% of peers support UT_PEX, 15-20% of peers support AZ_PEX and only about 5% of peers support BC_PEX. Peers supporting those three types of PEX protocols account for 95% of all the peers. Due to the prevailing usage of UT_PEX, we focus our attention on the measurement of UT_PEX.

Torrent ID	UT_PEX	AZ_PEX	BC_PEX
Torrent 1	70.2%	19.8%	4.9%
Torrent 2	65.9%	22.7%	2.8%
Torrent 3	75.9%	13.7%	7.8%
Torrent 4	74.3%	15.9%	6.9%
Torrent 5	79.7%	15.3%	2.5%

TABLE I: Distribution of BitTorrent Clients

Our passive measurement of PEX is based on a modified version of the LH-ABC 3.3.0.1 program [7], which is an extended version of ABC BitTorrent client and supports the UT_PEX protocol. The program was modified to record all the PEX messages, connections to and from other peers, and the download and upload rates of the client. No modification was made on other parts of the program.

A. Impact of PEX on Download Time

Our first experiment is to evaluate how the PEX impacts the download efficiency of a BitTorrent client. We randomly selected 100 torrents in different sizes, which were in the steady state [1] at the time of our experiments, from two top torrent-discovery sites - *PirateBay* [8] and *MiniNova* [2].

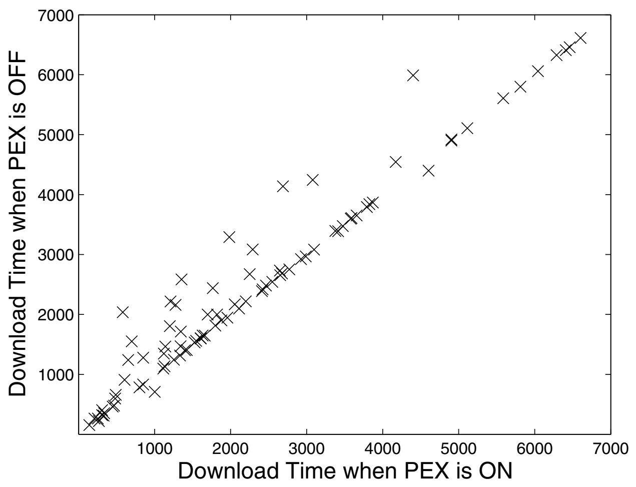


Fig. 1: Download time when PEX is ON vs. Download time when PEX is OFF

For each torrent, we performed downloads on two machines with the same OS and network configuration in parallel. These two machines are located at the same location but in different subnets. By turning the *pex_allowed* option on and off, we can control whether to allow peer exchange or not. In the experiment, we turned on the *pex_allowed* option of the BitTorrent

[1]Steady state refers to the state where the torrent size doesn't change radically during the experiments.

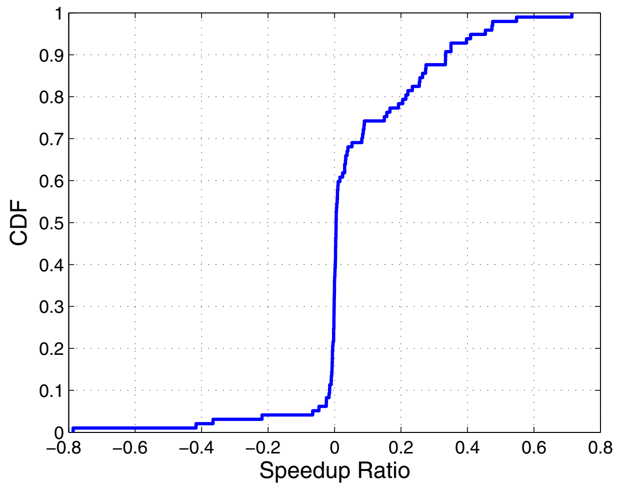


Fig. 2: CDF Distribution of the Speedup Ratio of all the tested torrents

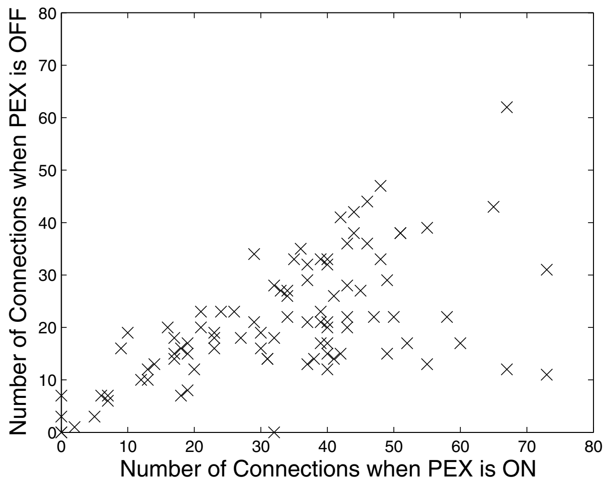


Fig. 3: Number of connections at the 5th minute with PEX ON vs. Number of connections at the 5th minute with PEX OFF

client on one machine and turned off the *pex_allowed* option on the other machine. The torrent download was started at the same time on two machines. Meanwhile, we also installed a third-party software PeerGuardian [9] on each machine to block data exchange between these two machines. The comparison of the download time with PEX on and off is shown in Figure 1. It is a scattering plot, in which each dot represents a torrent with its x-axis and y-axis value being the download time with PEX on and off respectively. From the figure, we can observe that the download time of some torrents can be significantly reduced when PEX is allowed. However, for a majority of torrents, the reduction of download time seems to be slight.

For a better understanding, let us define the *Speedup Ratio* as

$$\textit{Speedup Ratio} = \frac{T_d \text{ with PEX off} - T_d \text{ with PEX on}}{T_d \text{ with PEX off}}$$

where T_d is the download time of a torrent.

Among the tested 100 torrents, the mean value of speedup ratio is 0.07, which means a reduction of 7% of the download time when PEX is allowed. We further plot the CDF distribution of speedup ratio of tested torrents in Figure 2. It clearly shows that about 40% of the torrents have a speedup ratio greater than 0. Among the left torrents, 50% of the torrents have a speedup ratio of 0, which implies that PEX is not helpful for those torrents.

We also recorded the number of connections of our instrumented BitTorrent clients at different time slots. Figure 3 compares the number of connections at the 5-th minute when PEX is on and off. The figure clearly shows that PEX can enrich the connectivity of the client greatly, which can help the client reduce its download time.

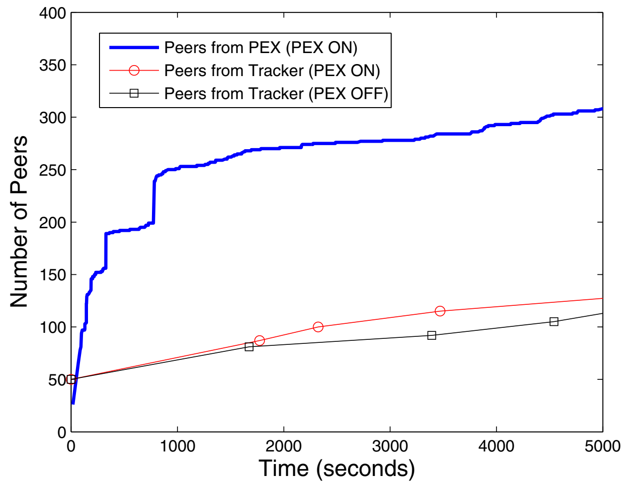


Fig. 4: Number of peers obtained from PEX and from the tracker

To further understand how PEX reduces the download time of a portion of torrents, we analyzed their download traces in details. As the analysis results are similar, we only present the results of one typical torrent here.

Figure 4 plots the number of unique peer addresses received from PEX and the tracker. From the figure, we observe that the client can quickly learn a large number of peer addresses through PEX after initially contacting the tracker. After joining the torrent for 15 minutes (900 seconds), the client obtained about 250 unique peer addresses. To our surprise, for the LH-ABC client, the use of PEX doesn't reduce the number of queries sent to the tracker. In the first 5000 seconds, when PEX is on, the client sent as many queries (i.e., 4 queries) to

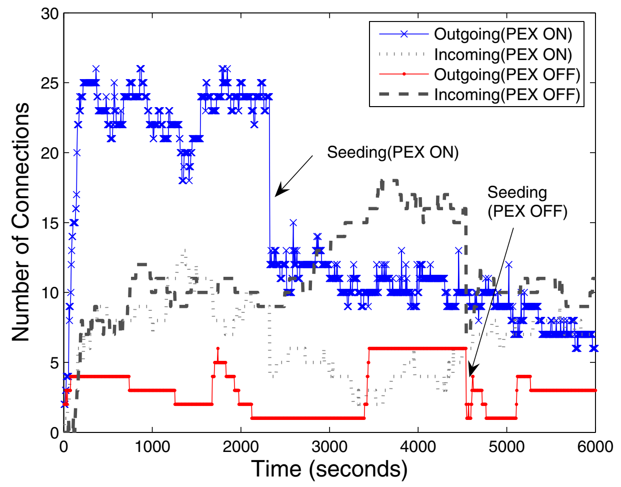


Fig. 5: Incoming and outgoing connections when PEX is on and off

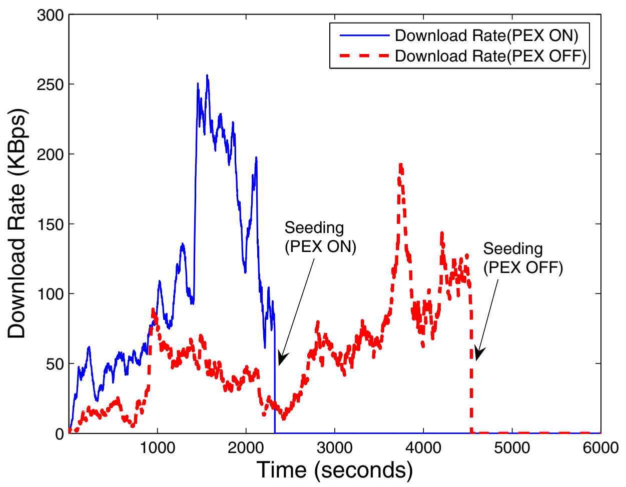


Fig. 6: Download rate when PEX is on and off

the tracker as when PEX was off. Even when the client enters the seeding state, it continues to send queries to the tracker. This observation is somewhat contrary to one of the design objectives of PEX, namely reducing the load on the tracker.

Figure 5 depicts the incoming and outgoing connections when PEX is on and when it is off. From the figure, we find that the client can quickly establish about 25 outgoing connections with other peers when PEX is used. On the contrary, the client only has about 5 outgoing connections when PEX is not used. Although it is not necessarily true that more connections imply higher download rate, the client with more connections has more opportunities to be optimistically unchoked by other peers. When PEX is on, after staying in the torrent for about 2200 seconds, there is a clear drop in the number of outgoing connections. This is because the client has completed the download of the file and has released half of the outgoing connections.

The download rate of the client is shown in Figure 6. We can observe from the figure that, during the period $[0, 1000]$, even when PEX is used and the client has many outgoing connections (see Figure 5), the download rate is not very high. But once the client has received a sufficient number of chunks, it can trade chunks with more peers, and the download rate is increased rapidly.

In summary, PEX can help the client quickly acquire a large number of peer addresses in the torrent and thus establish more connections with other peers. With more connections, it is possible for the client to obtain more unchoked upload slots, which in turn speeds up the download rate accordingly.

B. *Freshness and Redundancy of PEX Information*

In this section, we conducted an experiment to evaluate the freshness of the information in PEX messages. In the experiment, we implemented a tracker crawler, which obtains the complete peer list of a given torrent every 2 minutes. It enables us to obtain multiple snapshots of the peer list in the torrent. Let $T(t_i)$ be the set of peers obtained from the tracker at time t_i $(i = 1, 2, ...)$ with $t_1 = 0$ second and $t_{i+1} = t_i + 120$ seconds.

Suppose the client receives a PEX message at time t, which satisfies $t_{i-1} < t \le t_i$. Let $P(t)$ be the set of peers contained in the message. We define the *Freshness Ratio* of the PEX message as below:

$$\text{Freshness Ratio} = \frac{|P(t) \cap T(t_i)|}{|P(t)|}$$

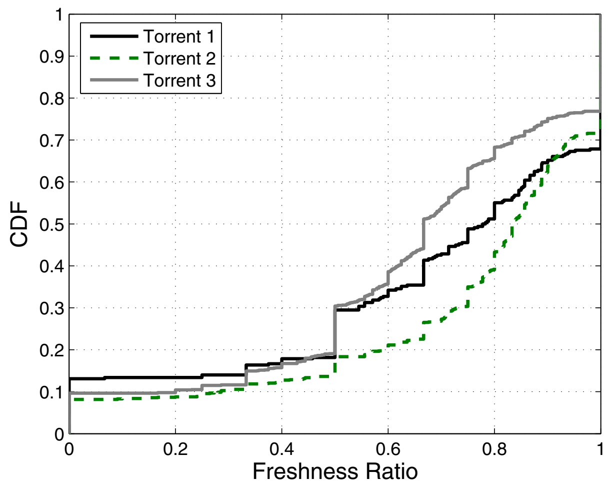


Fig. 7: CDF Distribution of the freshness ratio of all the PEX messages

The intuition behind the above metric is that, when a peer in the PEX message also appears in the peer list obtained from

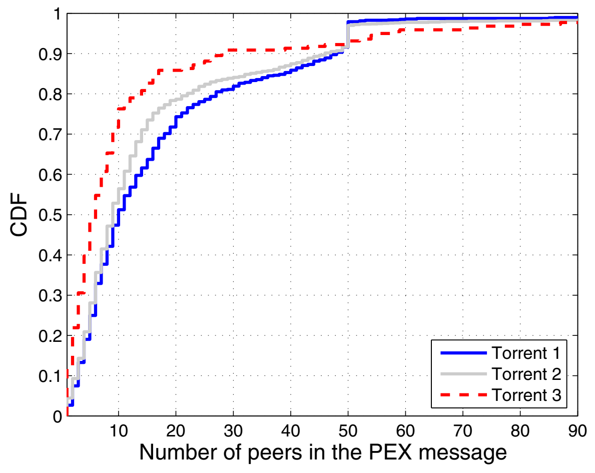


Fig. 8: Distribution of the number of peers in PEX messages

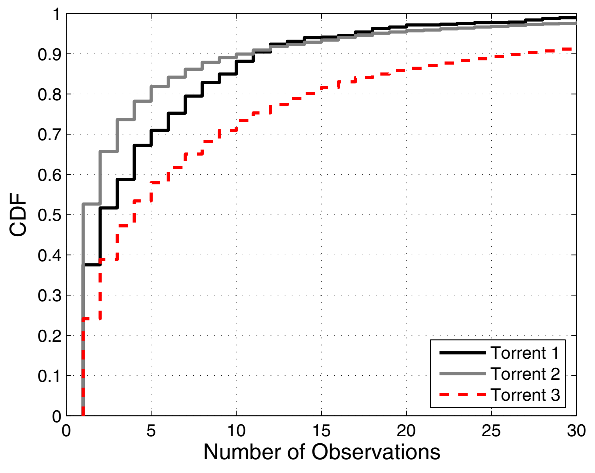


Fig. 9: Distribution of the number of repeated peer observations

the tracker crawling immediately after t, it is highly possible that the peer is a live peer. It also indicates to what degree PEX can replace the role of the tracker. Indeed, if all the peers contained in the PEX messages also appear in the peer list obtained from the tracker, there is no need to contact the tracker for peer lists after initial bootstrapping.

We conducted experiments for three torrents in different sizes. Figure 7 provides the distribution of freshness ratio of all the received PEX messages in three tested torrents. From the figure, we observe that about 20-30% of PEX messages are completely fresh and about 10% of PEX messages have a freshness ratio of zero. Over 80% of PEX messages have a freshness ratio greater than 0.5. Thus, PEX can potentially replace the role of the tracker in discovering new peers.

Figure 8 plots the distribution of the number of peer addresses contained in the PEX messages. 80-90% of PEX messages contain less than 30 peer addresses and less than 10% of PEX messages contain more than 50 peer addresses. According to the UT_PEX specification [5], the maximum number of peer addresses that can be put into a PEX message is 50. However, we observe that some BitTorrent clients do not follow the specification. In the log files, we even find that several PEX messages contain over 300 peer addresses.

For all the received PEX messages, we also calculate the number of repeated peer observations for the same peer and show the results in Figure 9. It is found that 20-40% of peers were observed more than 5 times. This indicates that there exists a large degree of redundancy during peer exchange.

To understand the message overhead incurred by peer exchange, using the traces obtained in the first experiment, we plot the CDF of the number of PEX messages received in the first hour for all 100 torrents in Figure 10. The number of received PEX messages varies greatly for different torrents, ranging from 0 to 1476. However, even considering the maximum value, the client only received 0.41 PEX message

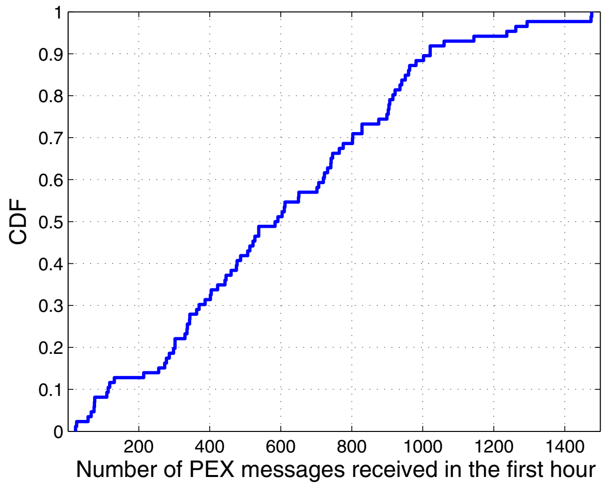


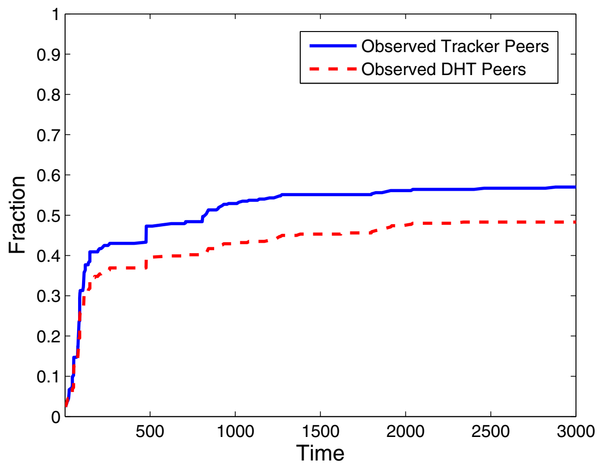
Fig. 10: CDF of the number of PEX messages received in the first hour

per second on average, which is not a heavy burden for the client.

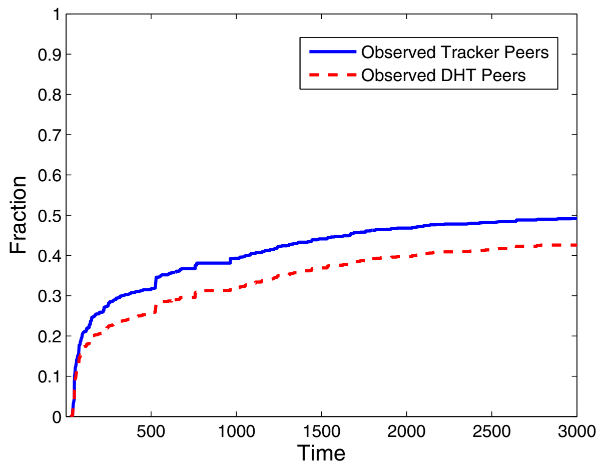
In summary, the fresh ratio of PEX messages is pretty high and PEX can potentially provide the functionality of the tracker after bootstrapping. There also exists a large degree of redundancy of PEX information during peer exchange, and the message overhead of PEX is not very high.

C. *Spread of PEX Information*

In this section, we evaluate how fast the peer information can spread via peer exchange. In the beginning of our experiment, we crawl both the centralized tracker and DHT to obtain two sets of peers. Define T_0 and D_0 be the set of peers obtained from the tracker and DHT respectively.



(a) small torrent



(b) large torrent

Fig. 11: Fraction of observed peers (a) small torrent; (b) large torrent

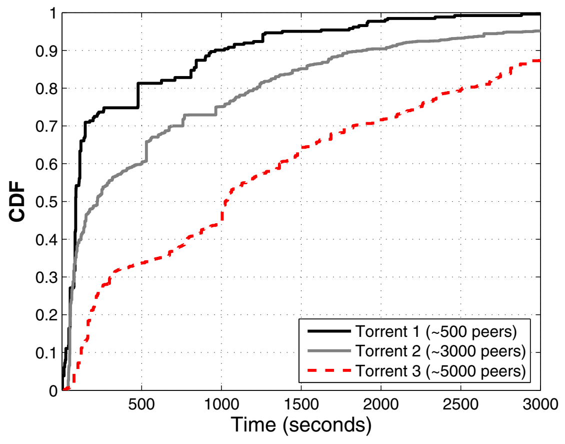


Fig. 12: Distribution of the first observation time of peers

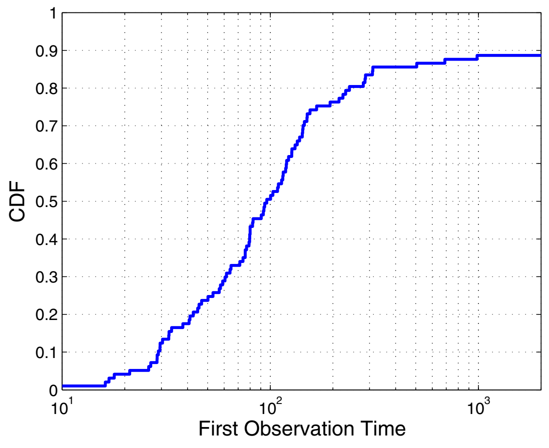


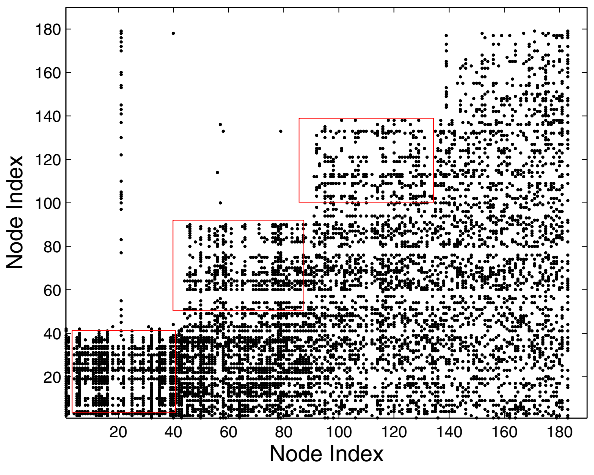
Fig. 13: CDF of the time to observe the IP of another machine

In Figure 11, we plot the fraction of peers in T_0 (D_0) that were also observed from PEX messages. For both small (with $\sim$ 500 peers) and large torrent (with $\sim$ 5000 peers), we find that it is difficult for the client to learn all the peers in T_0 and D_0 within a short period. In the very beginning, the client can quickly obtain about 30-40% of the peers in T_0 and D_0. But the fraction increases slowly with time. Observe that the client discovers less than 60% of peers in T_0 and D_0 even after running for over 3000 seconds.

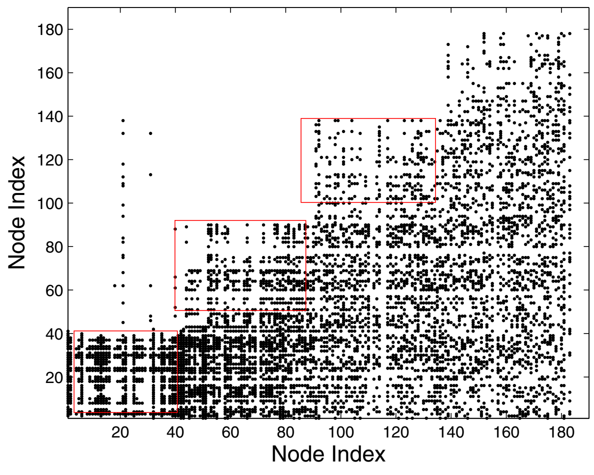
Possible reasons include: (1) some peers in D_0 or T_0 have already left the torrent. (2) the normal frequency of sending out PEX messages to neighboring peers is once per minute, which is not very frequent; (3) in the PEX messages, the client only propagates connected peers to its neighbors, instead of all the received peers. This improves the freshness of PEX information, but hinders the rapid spreading of peer information.

For all the observed peers in $T_0 \cup D_0$, we plot the distribution of the first observation time of each peer in Figure 12. We find that for small-size torrents, most of the peers were observed within a short period, e.g., for Torrent 1, 70% of all the observed peers were learned by the client within 200 seconds. However, for large-size torrents, it may take quite a long time to observe a peer in the set $T_0 \cup D_0$.

Using the 100-torrent traces obtained in the first experiment, we also performed another analysis about the spread of PEX information. In the first experiment, we have two machines that joined the torrent at the same time, with one machine equipped with PEX capacity. We check the traces of PEX messages to



(a) PEX ON



(b) PEX OFF

Fig. 14: Connectivity matrix when PEX is on and off

see how long it takes for the machine with PEX on to learn the address of another machine via PEX messages from other peers. The CDF distribution of the time to observe the address of another machine is plotted in Figure 13. Among all the 100 tested torrents, the machine can learn the address of another machine within 100 seconds for 50% of torrents, and within 2000 seconds for 90% of torrents. For the left 10% torrents, the machine cannot detect the address of another machine in the whole experiment period.

To summarize, we observe that the spread of PEX information is not as fast as we expected. It is difficult for the client to observe all the other peers in the torrent within a short period.

D. Impact of PEX on Topology Properties

To study the impact of PEX on the BitTorrent overlay, we create our own torrent on the Planetlab testbed. We installed LH-ABC clients on 200 Planetlab nodes and set up our own tracker using BitTornado. The data file to be distributed among peers is 500 MB. Instead of letting nodes join randomly, we let the 200 nodes join the torrent in batches. In the beginning, 50 nodes join the torrent; every 10 minutes, we let 50 more nodes join the torrent until all the nodes have joined. The experiments were run for about 2 hours. We downloaded all the logs after the experiments were completed. [2]

Figure 14 presents the connectivity matrix of peer connections 5 minutes after all the peers join the torrent. A point (x, y) on the plot indicates that peer x is connected to peer y. Node indexes are sorted in ascending order based on their joining time.

[2]Some PL nodes have difficulty in connecting with the tracker. These nodes were removed during analysis.

From the figure, we can see that PEX enriches the connectivity among peers, especially in the regions within the boxes. However, the increase is not as significant as we expected. This is possibly because the number of Planetlab nodes is limited, and all the Planetlab nodes have public reachable IPs.

Even without PEX, the client can still discover enough reachable peers and establish connections with them. In case that the torrent is dynamic and nodes are heterogeneous, the peer list returned by the tracker would contain a larger fraction of stale or NATed/firewalled peers. In those cases, simply by querying the tracker once in the beginning, it is difficult for the peer to establish enough outgoing connections. With peer exchange, the client is able to discover more reachable peer addresses.

Moreover, unlike the simulation experiments in [3], we never observed the formation of chain-like topologies in our experiments.

IV. Related Work

In recent years, there have been extensive studies on the measurement of BitTorrent. Izal et al. [10] analyzed the behavior of a single torrent over a five-month period. Pouwelse et al. [11] investigated the torrents hosted by Supernova. Guo et al. [12], [13] measured torrent evolution, service availability, and client performance in BitTorrent by analyzing a number of tracker traces from [14] and torrent file download traces. Neglia et al. [15] investigated the availability of BitTorrent systems by collecting about 22,000 torrents from two torrent-discovery sites and mainly focused on tracker reliability issues. Legout et al. [16], [17] studied the peer selection strategy in BitTorrent and observed the clustering of similar-bandwidth peers. Dale et al. [18] investigated the topology properties of the BitTorrent overlay via Planetlab experiments.

There have also been several measurement studies on DHTs. In [19], [20], [21], Steiner et al. performed a measurement study of the KAD DHT to investigate the characteristics of its peers. In [22], Falkner et al. crawled the Azureus DHT and provided characterizations of churn, overhead, and performance and also proposed a modified DHT lookup algorithm. Stutzbach et al. [23] measured the lookup performance of KAD DHT used by eMule file-sharing systems.

However, there have been very few studies on the peer exchange feature in BitTorrent. To our knowledge, the only related work is [3], in which Al-Hamra et al. used a simulation based approach to analyze the impact of peer exchange on the overlay topology of BitTorrent.

Our paper differs in that, instead of using simulation, we perform Internet measurements and Planetlab experiments to examine the behavior of PEX in operational systems. To the best of our knowledge, this is the first detailed measurement study of the impact of peer exchange in real swarms using real client implementations.

V. Conclusion

In this paper, we have presented an extensive passive measurement and Planetlab-based experimental study of the BitTorrent Peer Exchange (PEX) protocol. We have studied the impact of PEX on download time and analyzed the PEX and connection logs closely. Our results indicate that PEX enables peers to quickly acquire a large number of peer contacts, and reduce the download time by 7% on average. We also designed experiments to evaluate the freshness, redundancy and spread of PEX messages. We found that, over 80% of PEX messages have a freshness ratio greater than 0.5, but there exists a large degree of redundancy in PEX messages. We also observed that the spread of PEX messages is not very fast. In particular, PEX does not allow a peer to discover all other peers in a short time. Finally, by using the Planetlab platform, we examined the impact of PEX on the BitTorrent overlay topologies. We found that PEX can enrich the connectivity of BitTorrent overlay, but we didn't observe the chain-like topology reported in [3].

This paper presents an initial study of BitTorrent's PEX protocol. In the next step, we plan to perform a more complete measurement study. We will test how useful PEX is under a high-churn environment, and understand how PEX impacts other graph parameters of BitTorrent topologies, such as the diameter and other spectral moments.

Acknowledgements

This work has been in part supported by NFS grant CNS-0917767, the Fundamental Research Funds for the Central Universities, Sun Yat-Sen University (grant no. 2009-35000-3161425/09LGPY56) and Sun Yat-Sen University "Hundred Talents Program" (grant no. 35000-3226138). Xiaojun Hei's research work was supported by the National Science Foundation of China under Grant No. 60972014, the Technology Support Plan of the National 'Eleventh Five-Year-Plan' of China under Grant No. 2006BAK11B04-3 and Grant No. 2006BAJ07B06-3.

References

[1] "Mininova's Torrent Downloads Double to 7 Billion in a Year," http://torrentfreak.com/mininovas-torrent-downloads-doubled-in-a-year-090105.
[2] "MiniNova," http://www.mininova.org/.
[3] A. Al-Hamra, A. Legout, and C. Barakat, "Understanding the Properties of the BitTorrent Overlay," in *Technical Report, INRIA, Sophia Antipolis*, 2007.
[4] "AZ PEX protocol," [Online]. Available: http://www.azureuswiki.com/index.php/Azureus_messaging_protocol.
[5] "UT PEX protocol," [Online]. Available: http://www.rasterbar.com/products/libtorrent/extension_protocol.html.
[6] "Wikipedia - Peer Exchange ," [Online]. Available: http://en.wikipedia.org/wiki/Peer_exchange.
[7] "LH-ABC BitTorrent Client," [Online]. Available: http://code.google.com/p/lh-abc/.
[8] "The Pirate Bay," http://thepiratebay.org.
[9] "PeerGuardian," http://phoenixlabs.org/pg2.
[10] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Hamra, and L. Garces-Erice., "Dissecting BitTorrent: five months in a torrent's lifetime," in *Passive and Active Measurements*, Antibes Juan-les-Pins, France, Apr. 2004.
[11] J. A. Pouwelse, P. Garbacki, D. H. Epema, and H. Sips, "The BitTorrent P2P file-sharing system: Measurements and analysis," in *Proc. of International workshop on Peer-To-Peer Systems (IPTPS)*, Ithaca, NY, Feb. 2005.
[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of bittorrent-like systems," in *Internet Measurement Conference (IMC 05)*, Berkeley, California, USA, Oct. 2005.
[13] L. Guo, S. Chen, Z. Xiao, E. T. X. Ding, and X. Zhang, "A Performance Study of BitTorrent-like Peer-to-Peer Systems," in *IEEE Journal on Selected Areas in Communications (JSAC), Vol. 25, No. 1, pp. 155-169*, 2008.
[14] A. Bellissimo, B. N. Levine, and P. Shenoy, "Exploring the use of bittorrent as the basis for a large trace repository." in *Tech. Rep. 04-41, University of Massachusetts Amherst*, Jun. 2004.
[15] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems," in *IEEE INFOCOM*, 2007.
[16] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in bittorrent systems," in *Proc. ACM SIGMETRICS*, San Diego, CA, Jun. 2007.
[17] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest First and Choke Algorithms Are Enough," in *ACM SIGCOMM/USENIX IMC*, 2006.
[18] C. Dale, J. Liu, J. Peters, and B. Li, "Evolution and Enhancement of BitTorrent Network Topologies," in *IEEE IWQoS'08*, 2008.
[19] M. Steiner, T. En-Najjary, and E. W. Biersack, "A Global View of KAD," in *Proc. of ACM Internet Measurement Conference (IMC)*, 2007.
[20] M. Steiner, D. Carra, and E. W. Biersack, "Faster Content Access in KAD," in *IEEE Conference on Peer-to-Peer Computing (P2P)*, 2008.
[21] M. Steiner, T. En-Najjary, and E. W. Biersack, "Long Term Study of Peer Behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, 2009.
[22] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson, "Profiling a Million User DHT," in *Proc. of ACM Internet Measurement Conference (IMC)*, 2007.
[23] D. Stutzbach and R. Rejaie, "Improving Lookup Performance over a Widely-Deployed DHT," in *IEEE INFOCOM'06*, 2006.