Topology-aware PEX for Improving BitTorrent

HyunYong Lee
The University of Tokyo
Tokyo, Japan
ifjesus7@gmail.com

Akihiro Nakao
The University of Tokyo
Tokyo, Japan
nakao@iii.u-tokyo.ac.jp

***Abstract*—In this letter, we propose one easily deployable traffic localization technique, *tPEX* for BitTorrent. Using peer exchange (PEX) of BitTorrent, tPEX injects a set of local peers to the local peers in the target network domain so that they realize the traffic localization. Through simulations, we show that, in AS with 200 simultaneous local peers, tPEX enables the local peers to download around 60% of content within the same AS. The traffic localization by tPEX also improves the download performance by 45%. Even in AS with 13 simultaneous local peers, tPEX enables the local peers to download around 35% of content within the same AS and improves the download performance by 17%.**

***Index Terms*—BitTorrent, peer exchange, topology-aware, traffic localization**

I. Introduction

The network-oblivious user matching in BitTorrent [1] is not network-efficient approach from both ISP's and users' perspectives. Considering extensive BitTorrent traffic volume [2], the network inefficiency caused by BitTorrent traffic is not trivial. The traffic localization (i.e., user matching within the same network domain) is one promising way to improve the network efficiency and satisfy both ISP (by reducing the inter-domain traffic volume) and users (by improving the download performance). Even though various approaches [3], [10], [4], [12] have been proposed to realize the traffic localization, most approaches require a modification of existing BitTorrent system, which makes a deployment difficult. For example, to apply application-layer traffic optimization (ALTO [3])-like approach, we need to install the central server that provides the better-than-random user matching guidance. We also need to modify P2P system so that the guidance can be reflected in selecting communication partners.

In this letter, we propose one practical approach to realize the traffic localization for BitTorrent. We exploit one feature of BitTorrent rather than depending on the modification of BitTorrent system. Nowadays, most BitTorrent clients (i.e., more than 95% [8]) support peer exchange (PEX). PEX enables each BitTorrent client to exchange the neighboring peer list with each other. Therefore, using PEX, we can inject local peers (i.e., the peers of the target network domain) to increase the probability of local communications within the target network domain. We call this approach as topology-aware PEX, or *tPEX*. tPEX requires BitTorrent user information such as IP address and port number. Thus, we first introduce one efficient crawling approach to collect BitTorrent user information. Then, we discuss possible ways to realize tPEX. Basically, the communication within the same network domain realizes the traffic localization. In addition, if tPEX injects some peers that are closer to the local peers than the neighboring peers of the local peers, we may increase the download performance of local peers, since smaller AS hop count usually means better communication performance. In BitTorrent, the user matching is affected by the content availability. Therefore, tPEX reflects AS hop count and content bitmap (showing availability of content pieces).

Through simulations, we show that tPEX achieves the traffic localization. Basically, AS with more local peers shows better performance improvement by tPEX. In AS with 200 simultaneous local peers, tPEX increases the portion of intra-AS traffic volume from 20% to 60% and improves the download performance by 45%. Even in ASes with small number of local peers, i.e., 24 and 13 simultaneous local peers at maximum, tPEX increases the portion of intra-AS traffic volume from 0% to around 50% and 35% and improves the download performance by 42% and 20%, respectively. One interesting observation in ASes with small number of local peers is that tPEX injecting the peers of m-hop ($m \geq 0$) ASes shows two times more intra-AS traffic volume than than tPEX injecting the local peers only by improving the content availability. But, tPEX reflecting content bitmap does not show noticeable improvement compared to tPEX with random selection, which means BitTorrent's rarest first policy is enough for high content availability. Above results show that tPEX can satisfy both ISP and users even in ASes with small number of local peers. Considering that tPEX just requires several machines and no modification is required, we argue that tPEX is easily deployable and practical technique.

II. Topology-aware PEX

In BitTorrent, the centralized server, called tracker, manages a list of peers sharing the same content (i.e., swarm) and returns certain number of peers randomly selected (e.g., 50) to newly joining peer to the swarm. The newly joining peer joins the swarm by making connections with the peers returned by the tracker. The network-oblivious user set by the tracker leads to poor network efficiency and suboptimal content distribution performance. In this environment, one obvious way to improve the network efficiency and the content distribution performance is to let local peers know other local peers so that the probability of local communications increases. Even though the tracker is at best position to do this by returning the topology-aware peer set [4], this approach requires the

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

modification of tracker. Instead, we propose tPEX that exploits PEX that is a de facto standard of BitTorrent to improve BitTorrent in terms of the network efficiency and the content distribution performance.

A. Crawling Peer Information

tPEX needs to contact every peer of the target swarms in the target network domain. We assume that ISP selects the target swarms according to its own rule (e.g., a number of local peers). For this, we need the peer information such as IP address and port number. To collect the peer information in BitTorrent, the crawler first collects *.torrent* file to know the tracker address written in .torrent file. Unlike most existing approaches that download and analyze the web pages to extract .torrent file [6], our crawler exploits a *.torrent storage server* (specialized in storing and redistributing .torrent files [7]). Our crawler first collects the identifiers of .torrent files by issuing *Scrape-ALL* (that returns all identifiers of .torrent files managed by the tracker) to existing trackers. Then, our crawler downloads .torrent files with the identifiers by accessing the torrent storage server (e.g., with *http://torrage.com/torrent/torrentID*).

With the collected .torrent files, an usual way to collect the peer information is to contact each tracker (described in .torrent file) periodically. Instead of contacting several trackers for the same content as most existing approaches do, our crawler contacts only one representative tracker (that is defined as the tracker that maintains the maximum number of peers in a swarm). 90% of swarms are managed by multiple trackers and the average number of trackers in each swarm is 4.82 [5]. Due to this reason, most existing approaches require several number of server machines and waste the resources to crawl redundant data. On the other hand, our crawler collects around 80% of all existing peers by contacting the representative tracker with only one server machine within one hour. Please refer to [5] for more information about our crawling scheme.

B. tPEX

PEX supports direct exchange of neighboring peer information between BitTorrent clients without depending on the tracker. Even though PEX conventions between BitTorrent developers [9] support the addition of newly joining peers and the removal of offline peers, the addition of newly joining peers is only allowed in practice due to the security reason. PEX message cannot be sent more than once a minute. Some BitTorrent versions ignore a PEX message including more than 10 peers. Thus, in this letter including the evaluation part, we assume that each PEX message includes 10 peers at maximum.

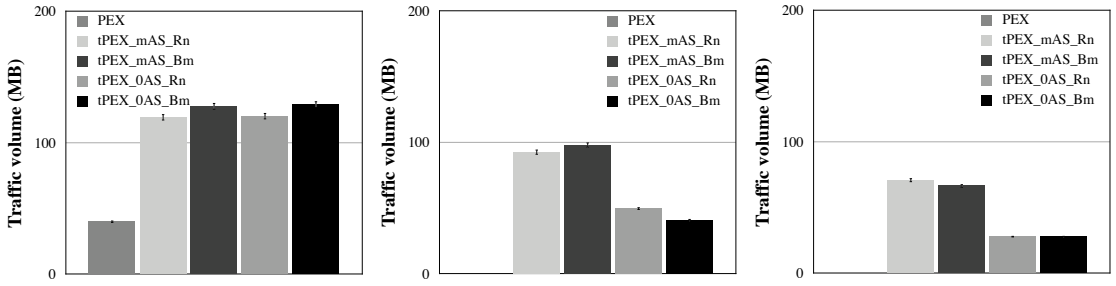
tPEX and PEX are different in selecting the peers to be sent in PEX message. PEX randomly selects the peers among the newly contacted peers. On the other hand, tPEX reflects two aspects: AS hop count and content bitmap (Algorithm 1).

AS hop count BitTorrent adopts tit-for-tat (TFT) strategy to select upload targets, since it has a limited number of upload slots (e.g., 4 is default). With TFT strategy, BitTorrent client prefers to upload a content to peers who have uploaded a content to itself in the past at high bandwidth. It means that peers that are close to each other in terms of AS hop count may prefer to communicate with each other, since the smaller number of AS hop counts usually mean the better communication performance. Therefore, if we inject the local peers, it may achieve the traffic localization. In addition, users may be able to improve the download performance further if they know some peers that are closer to themselves than the neighboring peers that they already know. Therefore, tPEX injects peers of (m+1)-hop ASes if $\sum_{i=0}^{m} N_i < 10$, where N_i is the number of peers in i-hop ASes. In any case, the peers in closer ASes are preferred.

Content bitmap In BitTorrent, the communication is also driven by the content availability. To increase the probability of communication within the same AS or across small number of AS hop counts, the peers to be sent in PEX message should have many pieces that can be downloaded by each target peer. Therefore, if the number of candidate peers in i-hop ASes (0 $\leq$ i) is more than k (remaining peer limits in PEX message, k $\leq$ 10), we reflect $BM_c - BM_t$, where BM_j indicates the content pieces that peer j has, t is the target local peer, and c is one of the candidate peers. Then, top-k peers in terms of $BM_c - BM_t$ are inserted into PEX message.

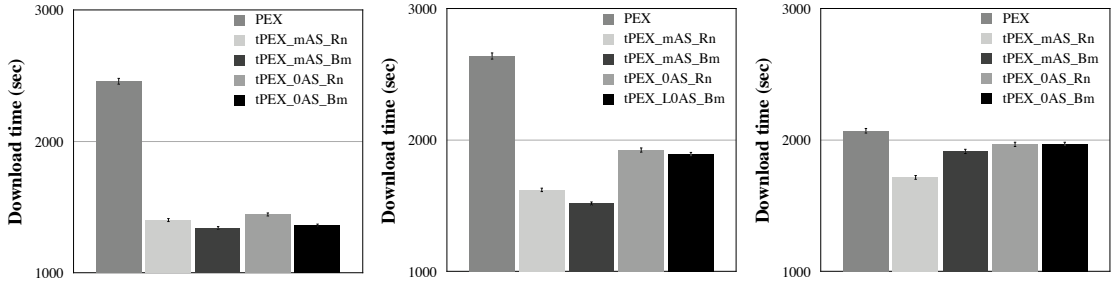
Algorithm 1 Peer Selection in tPEX

```
1:  max_injection = 10;
2:  P = get_peers_of_target_swarm();
3:  for each peer in the target network domain do
4:    injection_num = 0;
5:    I = { };
6:    while 1 do
7:      C = get_peers_of_smallest_AS_hop_count(P);
8:      if |C| ≤ max_injection - injection_num then
9:        I ← I + C;
10:       P ← P - C;
11:       injection_num += |C|;
12:     else
13:       calculate_bitmap_difference(C);
14:       descendig_order(C);
15:       for 0 ≤ i < max_injection - injection_num do
16:         I ← I + c_i;
17:         P ← P - c_i;
18:         injection_num ++;
19:       end for
20:     end if
21:     if injection_num == max_injection then
22:       break;
23:     end if
24:   end while
25:   send_pex_message(I);
26: end for
```



(a) AS1 (b) AS2 (c) AS3

Fig. 1. Average intra-AS traffic volume of each peer.



(a) AS1 (b) AS2 (c) AS3

Fig. 2. Average download completion time.

III. Evaluation

A. Simulation Setup

Before our simulation based on ns-2 [13], we crawled 2,802,269 swarms and conducted additional 24-hour peer-level measurement to know peer online time and type for 1,218 swarms with more than 1,000 peers. Finally, we chose one representative swarm that includes 4,909 peers that are distributed over 473 ASes. The tracker returns 50 peers and the maximum number of neighboring peers is 100. Each peer has different number of download limits (from 10 to 20, evenly distributed) and different amount of uplink capacity (100KB/s, 200KB/s, and 300KB/s, evenly distributed). But, each peer has the same downlink capacity, 300KB/s. 200MB-sized content (consisted of 800 chunks of 256KB) is shared. We set 50ms delay for communications within the same AS and communications with neighboring AS. PEX message is sent every 120 seconds. For performance comparison, tPEX injects the peers of m-hop ASes (mAS) or local peers only (0AS) and tPEX considers the bitmap difference (Bm) or not (i.e., random selection, Rn). Therefore, we have 4 different approaches, tPEX_mAS_Bm, tPEX_mAS_Rn, tPEX_0AS_Bm, and tPEX_0AS_Rn. We apply each tPEX approach to 3 ASes with different number of local peers. AS1/AS2/AS3 has 535/53/23 local peers in total during the simulation and has 200/24/13 simultaneous online peers at maximum. We repeat our simulations until the results converge. We show the average across the results together with the standard deviation.

B. Analysis

The average download completion time (Fig. 2) is closely related to the intra-AS traffic volume (Fig. 1). The more intra-AS traffic volume, the better download performance. Therefore, we focus on examining the traffic volume. Fig. 1 shows the average amount of intra-AS traffic volume that each peer generates to download 200MB-sized content. Without tPEX, only AS1 shows the intra-AS traffic volume. On the other hand, AS2 and AS3 do not show the intra-AS traffic volume, since they have too few local peers to see the local peers in the peer set returned by the tracker. But, once tPEX is applied, the intra-AS traffic volume increases noticeably. In AS1, the average number of local neighboring peers increases from 10 to 60 by tPEX. tPEX increases the intra-AS traffic volume of AS1 from 40MB to around 120MB. Even in AS2 and AS3 that have small number of local peers, with tPEX, around 100MB and 70MB of content is downloaded within the same AS, respectively. In AS2 and AS3, with tPEX, the average number of local neighboring peers is just 6 and 3, respectively. It means that tPEX can achieve the traffic localization even with small number of local peers. But, the total amount of reduced inter-AS traffic volume is proportional to the number of local peers. Above result also shows that tPEX application to AS with more number of local peers leads to more number of local neighboring peers, which leads to more amount of intra-AS traffic volume. But, the increase of the intra-AS traffic volume is not linear to the increase of the number of local peers in the target AS because of the limit of number of neighboring peers and the limit of aggregate uplink capacity.

tPEX_mAS_Bm and tPEX_0AS_Bm do not show noticeable difference in AS1, since only local peers are injected. AS1 has a large number of local peers and thus tPEX does not need to consider the peers of m-hop ASes for the injection. On the other hand, in AS2 and AS3, tPEX_mAS_Bm shows two times more intra-AS traffic volume than tPEX_0AS_Bm. The average number of local neighboring peers is almost same in tPEX_mAS_Bm and tPEX_0AS_Bm. But, the number of neighboring peers in 1-hop ASes is much different. With tPEX_0AS_Bm, the neighboring peers are somewhat evenly distributed over ASes of different hops. On the other hand, with tPEX_mAS_Bm, the neighboring peers in 1-hop ASes accounts more than half of the neighboring peers. A large number of neighboring peers in 1-hop ASes by tPEX_mAS_Bm allows the local peers to have the content pieces faster than tPEX_0AS_Bm and thus improves the content availability of the local peers. The improved content availability improves the local communications within the target AS.

tPEX_mAS_Bm shows slightly more intra-AS traffic volume than tPEX_mAS_Rn by improving the content availability. The slight difference means that BitTorrent's rarest first piece selection policy is a well-designed policy for improving the content availability. But, tPEX_mAS_Bm loses its benefit in AS2 and AS3. Actually, in AS2 and AS3, there is no difference between tPEX_mAS_Bm and tPEX_mAS_Rn, since the number of simultaneous online local peers is mostly less than 10. Therefore, we believe that tPEX_mAS_Rn is reasonable choice regardless of the number of local peers.

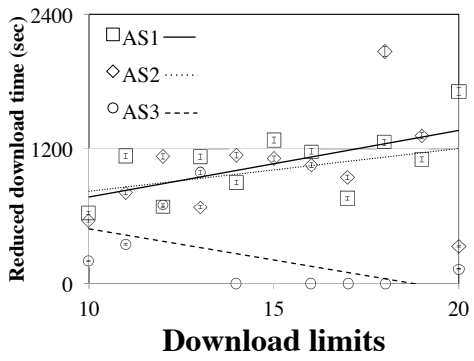


Fig. 3. Effect of the download limits on the performance improvement.

We also examine the relationship between the number of download limits and the improvement of download performance (Fig. 3). In AS1, the more number of download limits lead to better performance improvement, since more number of download limits means higher probability of more downloads. But, increasing the download limits may not affect the performance change if the downlink capacity is saturated (e.g., by high seeder/leecher ratio in large swarms). The more number of download limits do not always lead to better performance improvement in AS2 and AS3. This is due to that the aggregate uplink capacity of the local peers is too low to fill the download pipes. We do no have any relationship between the uplink capacity and the improvement of download performance.

IV. Related Work

Most existing ISP-driven and P2P-driven unilateral approach are not enough to satisfy both ISP and users. A widely used ISP-driven approach such as traffic shaping [11] usually degrades the P2P performance, which leads to the peer reactions such as data encryption and dynamic port change. Even though a few P2P applications [12] have developed technique to localize traffic based on the reverse engineering, there are fundamental limitations on what P2P can achieve alone. As a win-win approach, a bilateral approach based on the explicit collaboration between ISP and the peers has been proposed [10], [3]. In this approach, ISP provides certain type of guidance reflecting its network information and traffic engineering policy to the users. The users utilize the guidance to select their communication partners. The guidance based on the accurate network information enables the users to enjoy better download performance. At the same time, ISP can control the traffic as it wants. However, this approach requires a modification of existing P2P system, which makes a deployment difficult. On the other hand, tPEX is easily deployable win-win approach. With proper extension, tPEX may be able to reflect ISP's traffic engineering policy, since tPEX is ISP-driven approach.

V. Conclusion

The extensive penetration of PEX in BitTorrent opens a new way for the traffic localization of BitTorrent. In this letter, we propose tPEX to improve the traffic localization by injecting a set of local peers to the local peers in the target network domain. Our simulation results show that tPEX can reduce the inter-domain traffic volume and improve the content download performance noticeably, even in AS with small number of local peers. Based on our crawler, we plan to implement tPEX and test in real Internet environment.

References

[1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of P2PEcon*, 2003.
[2] SANDVINE Report, Global Internet Phenomena Report, http://www.sandvine.com/downloads/documents/Phenomena_2H_2012/Sandvine_Global_Internet_Phenomena_Report_2H_2012.pdf
[3] ALTO, http://datatracker.ietf.org/wg/alto/charter/.
[4] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in bittorrent via biased neighbor selection," in *Proc. of IEEE ICDCS*, 2006.
[5] M. Yoshida and A. Nakao, "A resource-efficient method for crawling swarm information in multiple bittorrent networks," in *Proc. of ISADS*, 2011.
[6] C. Zhang, P. Dunghel, D. Wu, and K. Ross, "Unraveling the bittorrent ecosystem," *IEEE Trans. on Parallel and Distributed Systems*, 2010.
[7] Torrage, Torrent storage cache, http://www.torrage.com/.
[8] D. Wu, P. Dhungel, X. Hei, C. Zhang, and K. W. Ross, "Understanding peer exchange in bittorrent systems," in *Proc. of IEEE P2P*, 2010.
[9] Peer exchange (PEX), http://en.wikipedia.org/wiki/Peer_exchange/.
[10] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Portal for (P2P) applications," in *Proc. of ACM SIGCOMM*, 2008.
[11] Cisco, Network-based application recognition, http://www.cisco.com/en/US/products/ps6616/products_ios_protocol_group_home.html.
[12] Kontiki, http://www.kontiki.com/.
[13] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/.