Approaches for Practical BitTorrent Traffic Control

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*—As practical ways to control BitTorrent traffic, we examine two basic approaches that exploit existing features of BitTorrent instead of modifying BitTorrent system. Based on PEX that allows BitTorrent clients to exchange their neighboring peer information directly, we propose *topology-aware PEX, tPEX* to inject some local peers to each local peer in the target network domain for the traffic localization. We also try to redirect the unavoidable inter-domain traffic from the transit links to the peering links by affecting tit-for-tat (TFT) strategy through the delay insertion (*indirectly guided TFT, gTFT*). Through simulations, we show that tPEX increases the intra-domain traffic volume by up to 316% and reduces the charging volume by up to 32.2% and the download completion time by up to 43.3%. gTFT reduces the inter-domain traffic volume of the transit links by up to 11.1% and increases the inter-domain traffic volume of the peering links by 9.2%. gTFT reduces the charging volume by up to 9.7%. Even though gTFT does not much affect the average download completion time, gTFT increases the performance difference among peers by up to 28.6% by adding the artificial delay selectively. tPEX+gTFT shows almost similar performance to that of tPEX, since gTFT loses its ability to redirect the inter-domain traffic once tPEX is applied. Above results show that tPEX can be the practical win-win approach to satisfy both ISP and users.**

I. Introduction

For successful P2P traffic control, any approach needs to satisfy users (who generate P2P traffic) as well as ISP (who wants to control P2P traffic). In this regards, existing unilateral approaches by ISP or P2P are not so practically satisfactory. A widely used ISP-driven approach is to use traffic shaping devices to rate limit P2P traffic [3]. This approach usually degrades P2P performance and leads to the user reactions such as encryption and dynamic ports to avoid being identified [19]. Even though there are some P2P-driven approaches to localize traffic based on reverse engineering [24], there is fundamental limitations what P2P achieve alone. In particular, it is almost impossible to reflect ISP's traffic control intention through the reverse engineering. As a win-win approach that satisfies both users and ISP, cooperative approach based on the collaboration between the users and ISP has been proposed [4]-[10]. In this approach, ISP provides some guidance reflecting its network information and traffic engineering policy to the users. Then, the users utilize the guidance when they select their communication partners. This approach allows the users to improve their content download performance based on the accurate network information. At the same time, ISP can control P2P traffic as it wants. But, this approach requires a modification of existing P2P system to reflect the guidance in selecting communication partners, which makes deployment difficult.

In this paper, we study practical win-win approaches for BitTorrent [1] that still accounts much portion of Internet traffic [22]). We exploit BitTorrent features instead of modifying existing BitTorrent system. One possible approach is to utilize peer exchange (PEX) that is supported by most BitTorrent clients (i.e., around 95% [2]). PEX allows BitTorrent clients to exchange their neighboring peer information with each other directly. Therefore, we can inject a set of local peers (i.e., the peers in the target network domain) to each local peer through PEX. The main goal of this *topology-aware PEX*, or *tPEX* is the traffic localization through the manipulation of neighboring peer set. Another possible approach is to affect tit-for-tat (TFT) strategy. With TFT strategy, BitTorrent client prefers to upload a content to peers who have uploaded a content in the past at high bandwidth. Therefore, if we can degrade download throughput through certain inter-domain link, a probability of communication through another inter-domain link may increase. To affect TFT strategy by degrading download throughput of specific inter-domain link, we add an artificial delay. The artificial delay is added to the inter-domain traffic of the transit links when the inter-domain traffic goes to a local peer who knows remote peers beyond both the transit and the peering inter-domain links. The main goal of this *indirectly guided TFT*, or *gTFT* is to redirect the unavoidable inter-domain traffic from the transit links to the peering links.

Through simulations, we examine tPEX, gTFT, tPEX+gTFT, and Delay that adds the delay to all inter-domain traffic of the transit links. tPEX increases the intra-domain traffic volume by up to 316% and reduces the charging volume by up to 32.2%. tPEX reduces the download completion time by up to 43.3% and the performance difference among peers (i.e., the standard deviation of download completion times of peers) by up to 76.6% by contacting every online local peer. gTFT reduces the inter-domain traffic volume of the transit links by up to 11.1% and increases the inter-domain traffic volume of the peering links by 9.2%. This result shows that gTFT can redirect some portion of the inter-domain traffic from the transit links to the peering links. gTFT also reduces the charging volume by up to 9.7%. Even though gTFT does not much affect the average download completion time, gTFT increases the performance difference among peers by up to 28.6% by adding the artificial delay selectively so that each local peer has different number of delayed inter-domain traffic. tPEX+gTFT shows almost similar performance to that of tPEX, since gTFT loses its ability to redirect the inter-domain traffic once tPEX is applied. Delay reduces the inter-domain traffic volume of the transit links by up to 48.8% and increases the inter-domain traffic volume of the peering links by up to 37.1%. Delay reduces the charging volume by up to 15.7%. But, Delay significantly increases the download completion time by up to 62.3% and the performance difference by up to 119.5% by adding the delay to all inter-domain traffic of transit links. Above results show that tPEX can be the practical win-win approach to satisfy both ISP and users. In addition, tPEX just requires several server machines for the swarm crawling and the manipulated BitTorrent client.

The rest of this paper is organized as follows. Section II introduces our approaches and Section III evaluates our proposed approaches. Section IV introduces related work and Section V concludes this paper.

II. Approaches

We try to control the inter-domain traffic caused by BitTorrent. In BitTorrent, there is the centralized server, called tracker that manages a list of peers sharing the same content (i.e., swarm). When a new peer wants to join an existing swarm, it first downloads .torrent file that is meta-data including the tracker address from a web site. The new peer contacts the tracker and the tracker returns certain number of peers (e.g., 50 is default) randomly selected among existing peers. The new peer joins the swarm by contacting each peer returned by the tracker. The topology-unaware random peer set by the tracker enables peers to communicate with other peers across network domains. In this environment, one obvious way for the traffic localization is to let local peers know other local peers by manipulating the neighboring peer set. Even with successful manipulation of neighboring peer set, there may be unavoidable inter-domain traffic, since BitTorrent communication is affected by a content availability. For example, if some chunks (i.e., fixed-sized segments of content) are not available from its local neighboring peers, a local peer needs to download the chunks from remote neighboring peers (i.e., the peers in other network domains) across network domains. In this case, if the remote neighboring peers are available through multiple inter-domain links, using cheaper inter-domain link may be preferred by ISP. Therefore, we may need to enable the local peers to use the inter-domain links that are preferred by ISP.

In this Section, we discuss two basic approaches to control BitTorrent traffic: tPEX for the traffic localization based on the manipulation of neighboring peer set and gTFT for the redirection of inter-domain traffic between inter-domain links based on the delay insertion. These functions depend on a crawling scheme in common, since they need peer information for their work. Therefore, we first introduce our crawling scheme. Then, we will describe each approach.

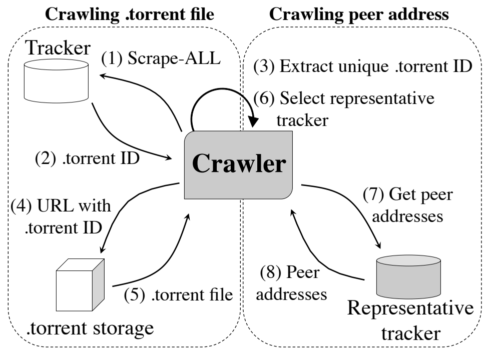


Fig. 1. Swarm-level crawling.

A. Swarm Crawling

The crawler collects peer addresses of existing swarms to identify target swarms (Fig. 1). Basically, there should be many local peers for the successful traffic localization [18]. We assume that each ISP has its own rule (i.e., specific number of local peers) to determine the target swarm based on the collected peer information.

Crawling .torrent files. Crawling swarm information requires *.torrent* file including the tracker addresses for certain content. With .torrent file, the crawler collects peer addresses by contacting the tracker. For collecting .torrent files, most existing approaches download web pages of BitTorrent torrent sites and analyze the web pages to extract .torrent files [14]. On the other hand, our crawler exploits a *.torrent storage server* (the server specialized in storing and redistributing .torrent files [25]) and *Scrape-ALL* request which returns all identifiers of .torrent files and number of peers managed by tracker. The crawler first collects the identifiers of .torrent files by issuing Scrape-ALL to existing trackers. Then, the crawler downloads .torrent files with the identifiers by accessing the torrent storage server (e.g., with *http://torrage.com/torrent/torrentID*).

Crawling peer information. With the crawled .torrent files, the crawler collects IP address and port number of peers to know the number of peers staying in the target network domain. Each peer clarifies the port number for BitTorrent when it registers itself to the tracker. An usual way to crawl the peer addresses is to contact each tracker (described in .torrent file) periodically. However, 90% of swarms are managed by multiple trackers and the average number of trackers in each swarm is 4.82 [15]. Due to this reason, the conventional approach wastes resources to crawl redundant data [14]. On the other hand, for each swarm, our crawler contacts only one representative tracker (that is defined as the tracker that maintains the maximum number of peers in a swarm) to efficiently obtain peer addresses. The crawler contacts the representative tracker periodically as the conventional approach does (i.e., the crawler repeats collecting a random subset until no more new peers can be discovered from the tracker). Our crawling scheme crawls around 80% of all existing peers by just contacting one representative tracker. In addition, our crawler broadens the crawling scope by obtaining peers behind NAT

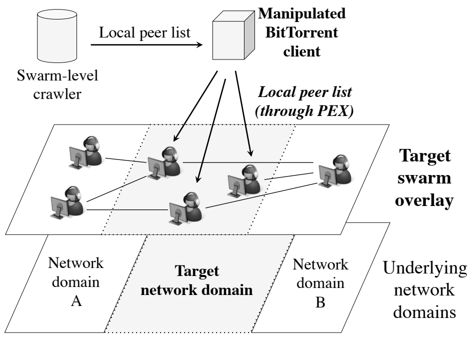


Fig. 2. Manipulation of neighboring peer set by tPEX.

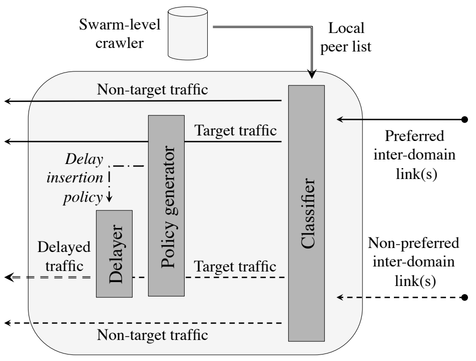


Fig. 3. Delay insertion for the redirection of inter-domain traffic by gTFT.

and firewalls. To achieve this, our crawler lets them connect to itself through actively advertising our crawler address to them. Finally, based on the collected swarm information, ISP determines target swarms according to its own rule. Please refer to [15], [16] for more information about our crawling scheme.

B. tPEX for Traffic Localization

tPEX manipulates the neighboring peer set of local peers for the traffic localization (Fig. 2). One obvious way to manipulate the neighboring peer set is to let the tracker return a set of peers based on topology information [17]. But, this approach requires a modification of existing tracker. Instead, we exploit PEX that allows peers to find new peers in the same swarm by exchanging their neighboring peer set with each other directly without relying on the tracker. Once the peer information of target swarm is given by the crawler, our manipulated BitTorrent client contacts each local peer and injects a set of local peers (randomly chosen among all identified local peers) through PEX.

C. gTFT for Redirection of Inter-domain Traffic

gTFT tries to affect TFT strategy based on the delay insertion to redirect the inter-domain traffic between inter-domain links. Every 10 seconds, with TFT strategy, BitTorrent client selects (i.e., *unchokes*) peers (up to certain number, e.g., 4 is default) to upload a content for next 10 seconds. For the peer selection, BitTorrent client cares about download performance from its neighboring peers during previous 10 seconds. BitTorrent client selects the peers who have uploaded content to itself in the past at high bandwidth. Thus, if we degrade download throughput of the non-preferred inter-domain links (from ISP's perspective) during ith unchoking interval, the degraded download throughput is reflected in the peer selection for (i+1)th unchoking interval and thus local peers may select peers beyond preferred inter-domain links. To redirect the inter-domain traffic from the non-preferred inter-domain links to the preferred inter-domain links, the artificial delay is added to the inter-domain traffic of non-preferred links when the inter-domain traffic goes to a local peer who knows remote peers beyond both the preferred and the non-preferred inter-domain links.

gTFT includes three components: traffic classifier, delayer, and policy generator (Fig. 3). Three components can be implemented by DPI tools [23] that support flow-level programmability such as packet dropping and adding delay. These components are attached to existing edge router so that they can handle the target traffic before being forwarded by the edge router.

Traffic classifier. The traffic classifier classifies given inter-domain traffic into the target traffic (i.e., the traffic from remote BitTorrent users to the local BitTorrent users in the target swarm) and the non-target traffic based on the peer information (i.e., IP address and port number) given by the crawler. The non-target traffic is just forwarded to the edge router. The target traffic is first forwarded to the policy generator so that the policy generator calculates the upload throughput of each remote peer to local peers. If the target traffic comes from the non-preferred inter-domain links, it is forwarded to the delayer for the delay insertion.

Delayer. The delayer adds the artificial delay to the target traffic according to the delay insertion policy given by the policy generator.

Policy generator. Given each target traffic, the policy generator updates the uploaded bytes from one remote peer (e.g., peer i) to one local peer (e.g., peer j, line 3 in Algorithm 1). Here, the policy generator only keeps the upload information of each remote peer during previous 10 seconds (line 1 in Algorithm 1), since TFT strategy considers the download throughput during previous 10 seconds. Even though the policy generator does not know when each local peer selects new peers for upload by applying TFT strategy, the policy generator calculates the average upload throughput during previous 10 seconds as a heuristic approach. Then, if the given target traffic comes from the preferred inter-domain links, the policy generator updates the minimum upload throughput (i.e., aggregate uploaded bytes) to the target peer (line 6 in Algorithm 1). If the target traffic from the non-preferred inter-domain link is given and its upload throughput to one local peer is higher than the minimum upload throughput to the same local peer (line 9 in Algorithm 1), the policy generator calculates amount of delay to be added. The goal of delay insertion is to degrade the upload throughput of the non-preferred inter-domain link

Algorithm 1 Delay insertion policy

```
1:  Every second: $T_{ij}[current\_time\_in\_sec\%10]$ = 0;
2:  Each given packet from remote peer i to local peer j:
3:      $T_{ij}[current\_time\_in\_sec\%10]$ += packet's payload size
4:
5:  if packet from preferred link then
6:      $M_j$ = find_min_throughput($j$)
7:  else if packet from non-preferred link then
8:      delay = 0;
9:      if calc_thpt($T_{ij}$) $>$ $M_j$ then
10:         for $1 \leq$ k $\leq 9$ do
11:             if calc_partial_thpt($T_{ij}$, (current_time - 9)%10, (current_time - k)%10) $<$ $M_j$ then
12:                 return (delay/delay_adjustment_factor)
13:             else
14:                 delay++
15:             end if
16:         end for
17:     end if
18:     return (delay/delay_adjustment_factor)
19: end if
```

lower than the minimum upload throughput. Therefore, based on the upload information of previous 10 seconds, the policy generator finds the point where the aggregate uploaded bytes (from 10 seconds before to the point) is lower than that of the minimum upload throughput case (line 10 and 11 in Algorithm 1). *delay_adjustment_factor* (line 12 and 18 in Algorithm 1) is used to limit the maximum amount of delay to be added. If there is no upload of the preferred inter-domain links to certain local peer, no delay is added to the inter-domain traffic of the non-preferred inter-domain links going to the local peer.

D. *tPEX+gTFT*

gTFT may be able to be combined with tPEX to handle the unavoidable inter-domain traffic. tPEX just injects a set of local peers while not removing existing remote neighboring peers from the neighboring peer set. According to PEX protocol specification, the removal of neighboring peers as well as the insertion of neighboring peer are possible. But, the removal is not supported by most existing BitTorrent implementations due to the security issue. For example, some malicious users can interrupt the content distribution by removing the normal users from the neighboring peer list. In addition, in order to avoid the swarm partitioning problem, each local peer may need to know some remote neighboring peers, even if we can remove some remote peers based on any approach. In this situation, peer communication is also affected by a content availability. The local peers sometimes need to download a chunk from remote peers when the chunk is not available from its local neighboring peers. In addition, all neighboring peers are not candidate peers to be selected based on TFT strategy every 10 seconds. Only peers who sent INTERESTED message are considered. A peer sends INTERESTED message to its neighboring peer that has the chunk that it does not have.

III. Evaluation

To examine the feasibility of the discussed approaches as the practical win-win approach, we build our simulation environments based on ns-2 simulator [26] as follows. We first crawled BitTorrent swarms before the simulation. With our crawler, we have the data of 2,802,269 swarms. After randomly selecting 1,218 swarms with more than 1,000 peers among the measured swarms, we have conducted peer-level measurement to know peer type and online time, and so on. Then, we finally applied our approaches to one representative swarm. The chosen swarm includes 6,493 peers and the peers are distributed over 684 ASes. We selected two ASes as the target AS. AS1 includes 720 local peers in total during the simulation time, 3 peering links, and 11 transit links. AS2 includes 535 local peers in total during the simulation time, 2 peering links, and 5 transit links. BitTorrent tracker returns 50 peers to a newly joining peer. 200MB-sized content (256KB-sized chunk) is shared. Each peer has the same link capacity (i.e., 300KB/s for download and 100KB/s for upload) and 5 upload slots including one optimistic unchoking slot. We set 50ms and 100ms delay for communication within AS and across ASes, respectively. We conduct the simulations up to 80,000 seconds.

As performance metrics, we use the traffic volume, the 95th percentile-based charging volume, and the download performance. For performance comparison, we first implement Vanilla where no approach is applied. In addition to tPEX, gTFT, and tPEX+gTFT, we implement Delay that adds 1 second delay to all inter-AS traffic of the transit links to compare gTFT with the simple delay insertion. In our simulation, PEX exchanges the information of 10 peers every 60 seconds and tPEX injects the local peers up to 10 every 120 seconds. In gTFT, the peering links are the preferred links. Even though we conducted the simulations of gTFT with the maximum delay of 2 seconds, 5 seconds, and 10 seconds, we do not have noticeable difference. Therefore, we only show the results of gTFT with the maximum delay of 2 seconds. We conduct each simulation 10 times and show the average across the results together with the standard deviation.

A. *Traffic Volume*

Fig. 4 and Fig. 5 show the average traffic volume of each peer. With tPEX, AS1 and AS2 increase the intra-AS traffic volume by 185% and 316% compared to Vanilla, respectively. In Vanilla, AS1 shows larger intra-AS traffic volume (i.e., 46.7MB) than AS2 (i.e., 32.8MB), since AS1 has more number of local peers. AS1 and AS2 have concurrent online local peers up to around 300 and 200 at certain simulation time, respectively. The difference in the number of local peers leads to different amount of intra-AS traffic volume in Vanilla. But, once tPEX is applied, AS1 and AS2 show similar amount of intra-AS traffic volume, 133MB and 136MB, respectively. AS1 and AS2 increase the total intra-AS traffic volume by

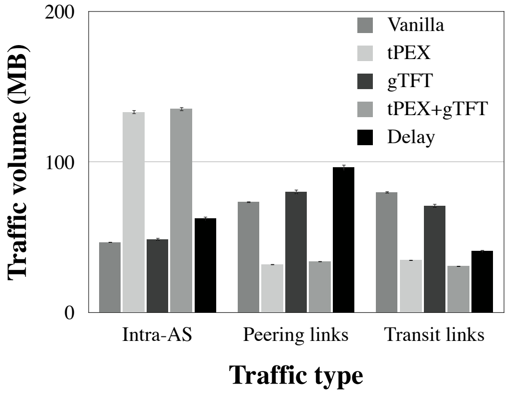


Fig. 4. Average traffic volume of each peer (AS1).

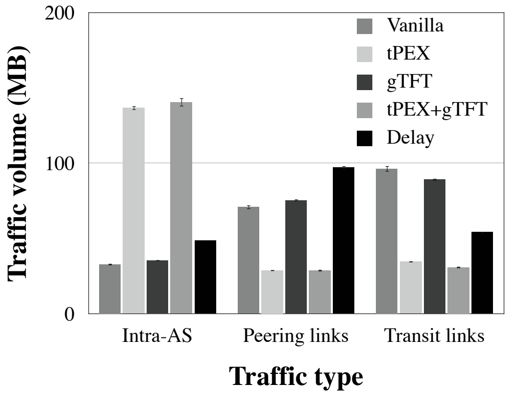


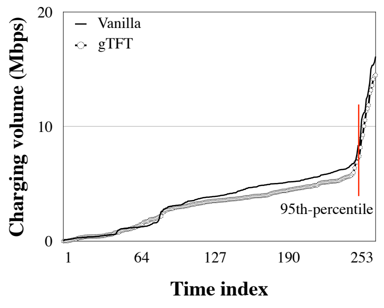
Fig. 5. Average traffic volume of each peer (AS2).

39.7GB and 44GB, respectively.[1] This may mean that the number of local peers does not have noticeable effect on the intra-AS traffic volume of each peer in tPEX when the number of local peers is larger than certain number. To examine the effect of the number of local peers on the intra-AS traffic volume by tPEX more, we examine one additional AS (i.e., AS3) that has 100 local peers in total during the simulation time and only 13 peering links. AS3 has around 50 concurrent online local peers at maximum. AS3 shows the intra-AS traffic volume of 7.2MB in Vanilla and 122.3MB with tPEX. AS3 increases the total intra-AS traffic volume by 7.5GB. This result first confirms that different number of local peers leads to different amount of intra-AS traffic volume in Vanilla, since more number of local peers mean higher probability to find other local peers when the tracker returns a set of peers randomly chosen. Once tPEX is applied, the difference in the number of local peers does not lead to significant difference in the intra-AS traffic volume, even though AS1 has concurrent online peers 6 times more than AS3. We conjecture that AS1, AS2, and AS3 with tPEX do not show noticeable difference in the intra-AS traffic volume, since the number of local peers is larger than the number of maximum active download limits (i.e., 20 in our simulation). The reason why AS3 shows smaller amount of intra-AS traffic volume than other ASes may be that AS3's aggregate upload capacity is lower than that of two other ASes. But, detailed study about the relationship between the performance improvement by tPEX and related factors including the download limits and the aggregate upload capacity is out of scope of this paper. The improved communication locality by tPEX leads to the reduction of inter-AS traffic volume of both the peering links and the transit links. This shows that tPEX is an effective way to increase the intra-AS traffic volume and reduce the inter-AS traffic volume.

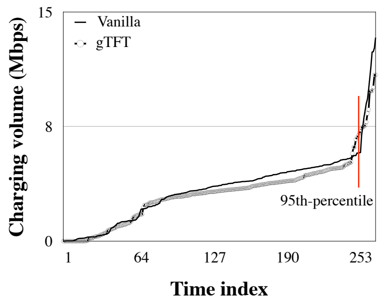
[1]Please note that Fig. 4 and Fig. 5 show the average traffic volume of each peer and the total traffic volume of AS is calculated as *the average traffic volume * number of peers*.

With gTFT, AS1 reduces the inter-AS traffic volume of the transit links by 11.1% (4.1GB in total) and increases the inter-AS traffic volume of the peering links by 9.3% (3.1GB in total). AS2 reduces the inter-AS traffic volume of the transit links by 7.5% (2.4GB in total) and increase the inter-AS traffic volume of the peering links by 6.2% (1.5GB in total). This result shows that gTFT's delay insertion redirects the inter-AS traffic from the transit links to the peering links. But, the reduced inter-AS traffic volume of the transit links is not fully redirected to the peering links. The remaining portion of the reduced inter-AS traffic volume of the transit links is redirected to the intra-AS one. With gTFT, AS1 and AS2 increase the intra-AS traffic volume by 0.9GB in total in both cases. Another observation is that larger number of local peers means larger amount of inter-AS traffic volume that can be redirected from the transit links to the peering links, since each peer downloads some portion of content from peers of other ASes. But once tPEX is applied, gTFT loses its ability to redirect the inter-AS traffic volume (or, there is not much traffic to be redirected), since tPEX already redirects much portion of inter-AS traffic into the intra-AS one. But, tPEX+gTFT still reduces the inter-AS traffic volume of the transit links based on the delay insertion. With tPEX+gTFT, AS1 and AS2 reduce the inter-AS traffic volume of the transit links by 2GB and 1.4GB in total compared to tPEX, respectively. AS1 increases the intra-AS traffic volume and the inter-AS traffic volume of the peering links compared to tPEX. But, AS2 just increases the intra-AS traffic volume by 1.4GB in total compared to tPEX. In AS2, gTFT does not redirect the inter-AS traffic from the transit links to the peering links when tPEX is applied. It means that we may not need to use tPEX+gTFT, since additional benefit by gTFT is not so much and tPEX is enough to decrease the inter-AS traffic volume.

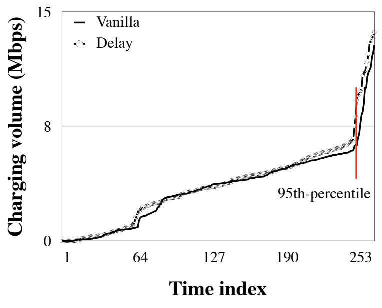
Delay shows similar result to that of gTFT. Like gTFT case, Delay reduces the inter-AS traffic volume of the transit links and the reduced inter-AS traffic volume of the transit links is redirected into the inter-AS traffic of the peering links and the intra-AS domain one. With Delay, AS1 reduces the inter-AS traffic volume of the transit links by 48.9% (17.9GB in total) and increases the inter-AS traffic volume of the peering links by 31.2% (10.5GB in total). AS2 reduces the inter-AS traffic volume of the transit links by 43.8% (14.2GB in total) and



(a) AS1 with gTFT



(b) AS2 with gTFT



(c) AS2 with Delay

Fig. 6. 5min-interval charging volume for inbound traffic.

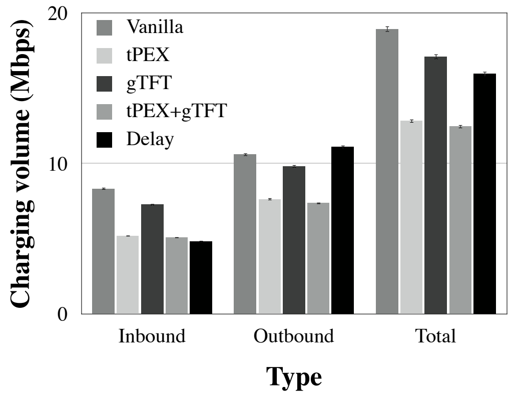


Fig. 7. Charging volume (AS1).

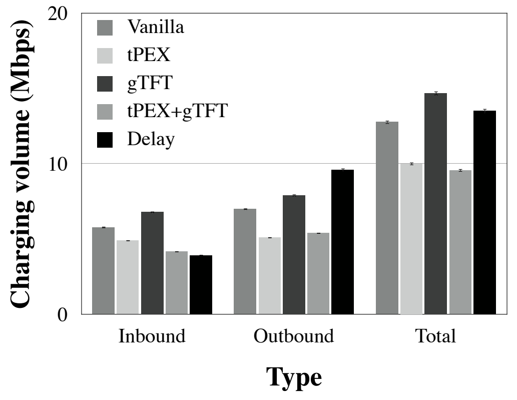


Fig. 8. Charging volume (AS2).

increases the inter-AS traffic volume of the peering links by 37.1% (8.9GB in total). AS1 and AS2 increase the intra-AS traffic volume by 7.3GB and 5.3GB in total, respectively. In terms of the traffic volume, Delay shows better reduction of inter-AS traffic volume and redirection of inter-AS traffic than gTFT.

B. Charging Volume

We calculate the charging volume based on the 95th-percentile method. Fig. 7 and Fig. 8 show the charging volume of inbound traffic, outbound traffic, and total traffic for AS1 and AS2, respectively. tPEX reduces the charging volume for both the inbound traffic and the outbound traffic. With tPEX, AS1 and AS2 reduce the total charging volume by 32.2% and 21.7%, respectively. Even though the reduction of the total charging volume is not so much for one swarm in our simulation (i.e., 6Mbps in AS1 and 2Mbps in AS2), large ASes that have many peers joining multiple swarms may have higher reduction of the charging volume. For example, we find that 1,218 swarms we crawled include 2,212,558 peers in total and these peers are distributed over 8,699 ASes. Among these ASes, 65 ASes include 50% of all peers. Thus, we guess that those large ASes may have noticeable reduction of the charging volume by tPEX's traffic localization.

gTFT reduces the charging volume for both the inbound traffic and the outbound traffic in AS1. With gTFT, AS1 reduces the total charging volume by 9.7%. But, gTFT increases the charging volume for both the inbound traffic and the outbound traffic in AS2 case. Even though gTFT reduces the total traffic volume of the transit links as we observed, gTFT's traffic redirection is not so successful in reducing the charging volume in AS2. To clarify this, we examine 5min-interval charging volume for the inbound traffic during our simulations (Fig. 6). For the sake of simplicity, in Fig. 6, we order the charging volume in ascending order. In both AS1 and AS2, gTFT shows lower charging volume than Vanilla case at most 5min-interval indices. But, unfortunately, in AS2 gTFT shows higher charging volume at the position that is used to calculate the 95th-percentile than Vanilla (Fig. 6(b)). That is why gTFT increases the charging volume for the inbound traffic in AS2 although it reduces the inter-AS traffic volume of the transit links. gTFT increases the charging volume for the outbound traffic due to the same reason. But, we guess that gTFT may reduce the charging volume for the inbound traffic and the outbound traffic in most cases as it does in AS1.

tPEX+gTFT shows slightly lower charging volume than tPEX. This is due to that gTFT reduces the charging volume (even though the reduction of the charging volume is not so much) in addition to tPEX's noticeable reduction of the charging volume. In tPEX+gTFT case, AS2 does not show the unfortunate case in reducing the charging volume.

Delay reduces the charging volume for the inbound traffic (i.e., by 41.9% in AS1 and 32.1% in AS2) and shows the lowest charging volume for the inbound traffic. But, Delay shows the largest charging volume for the outbound traffic. We examine 5min-interval charging volume for the outbound traffic of AS2 (Fig. 6(c)). Unlike gTFT case of AS2, Delay shows larger charging volume at most time indices than Vanilla. It means that Delay is not so effective to prevent the local peers from unchoking the remote peers beyond the transit links. Due to this reason, Delay increases the total charging volume of AS2 by 5.9%, since the increase of the charging volume for the outbound traffic is larger than the decrease of the charging volume for the inbound traffic. On the other hand, Delay reduces the total charging volume of AS1 by 15.9%, since the decrease of the charging volume for the inbound traffic is larger than the increase of the charging volume for the outbound traffic. This result shows that gTFT's selective delay insertion is better to affect TFT strategy than Delay's TFT-oblivious delay insertion.

C. Download Performance

Now, we give our attention to the download performance (Fig. 9 and Fig. 10). In Fig. 9 and Fig. 10, the value on top of the each column indicates the standard deviation of download completion times of peers while the height of column is the average download completion time. tPEX reduces the download completion time by 38.2% in AS1 and 43.3% in AS2, respectively. In addition, tPEX reduces the performance difference among peers. tPEX contacts every online peer every 120 seconds and thus most online peers can enjoy the improved communication locality. On the other hand, gTFT increases the download completion time by 6% in AS1. Even though gTFT increases the intra-AS traffic volume slightly in AS1 (Fig. 4), it does not lead to the improvement of download performance. The traffic volume of Fig. 4 is the sum of each traffic type during whole simulation time. Actually, gTFT decreases download speed per second by adding artificial delay to the inter-AS traffic of the transit links. Even worse, gTFT increases the performance difference among peers. In other words, gTFT's selective delay insertion degrades the download performance of some peers that are chosen for the delay insertion many times. For example, in AS1, gTFT's delay insertion happens 608,209 times in total during the simulation of 80,000 seconds. The average number of delay insertion for each peer is 1183 and the standard deviation is 708. The maximum and the minimum number of delay insertion for one peer is 5813 and 1, respectively. This result shows that each peer has different number of gTFT's delay insertion that degrades the download speed through the transit links and thus each peer shows different performance change by gTFT. Due to the traffic localization by tPEX, tPEX+gTFT shows similar download performance to that of tPEX. Delay increases the download completion time significantly (i.e., by 40.2% in AS1 and 62.3% in AS2). Even worse, Delay increases the performance difference among peers noticeably. This is due to that the number of connections across the transit links of each peer is different and thus each peer has different performance change by Delay's delay insertion.

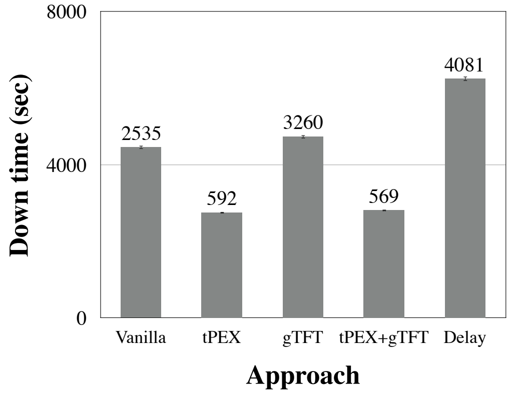


Fig. 9. Average download completion time (AS1).

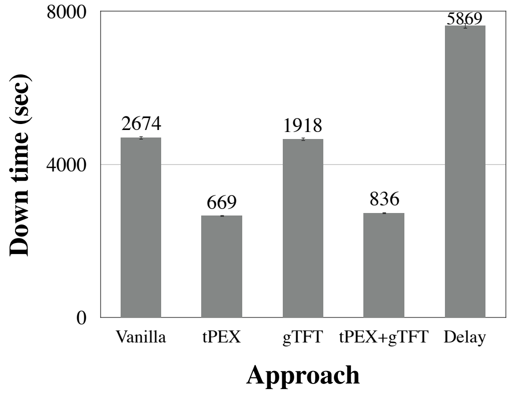


Fig. 10. Average download completion time (AS2).

Summary. tPEX is good to reduce the inter-AS traffic volume and the charging volume and improve the download performance. gTFT can redirect some inter-AS traffic from the transit links to the peering links. But, it is questionable that the reduction of charging volume by gTFT is larger than gTFT's installation cost. Even worse, gTFT degrades the download performance. Delay is simple and reasonable way to reduce the inter-AS traffic volume and the charging volume. But, it significantly increases the download completion time. Therefore, we can say that tPEX can be the practical win-win approach to satisfy both ISP and users.

IV. Related Work

Traffic shaping device [3], [21] is widely used to limit P2P traffic. This device usually degrades the P2P performance. Therefore, this unilateral control by ISP usually leads to the peer reactions such as data encryption and dynamic port change to avoid being identified. Even though a few P2P applications [24] have developed technique to localize traffic, there are fundamental limitations on what P2P can achieve alone. Recently, the enhanced P2P-driven ISP-friendly approach (aiming to satisfies both ISP and the peers) has been proposed. In [11], peers use CDN's DNS redirection technique for finding their communication partners that are likely to be closer to themselves. [12] tries to minimize the inter-domain cost and minimize the intra-domain cost by calculating an AS path between arbitrary two peers and using it as the guidance. In [13], they evaluate the impact of locality in the peer matching of BitTorrent on inter-domain links traffic and peers download completion time. But, they still depend on the reverse engineering that shows various fundamental limitations (e.g., ignorance about the ISP's traffic control intention).

The bilateral approach based on the explicit collaboration between ISP and the peers has been proposed to satisfy both entities. [4] introduces custom trackers and tracker interfaces to provide the peer selection guidance. In this approach, each peer has PID that may represent its network position like AS. The pDistance indicates the distance between a pair of PIDs and is used as the guidance. The basic idea of [5] and [6] is to provide a list of ordered peers or paths according to the predefined criteria. When a peer sends a list of possible neighbors to ISP, ISP ranks them according to certain criteria such as high bandwidth links. In [8], they show ISP-friendly technique is hard to achieve under real environment where peers are non-uniformly distributed. Then, they propose refinements of current proposals, allowing all users of P2P networks to be sure that their application performance is not reduced. In [7], the authors show that transmission cost of P2P streaming with ALTO [10] guidance can be reduced. They also show that ISP has to be careful not to over-localize traffic, for particularly delay-sensitive applications.

Rather than depending on the explicit user participation that requires the modification of existing P2P systems, we exploit existing features of BitTorrent to build the practical approach. We also try to satisfy both ISP and the users to propose the win-win approach.

V. Conclusion

Any approach needs to satisfy both ISP and the users to be successful in controlling P2P traffic. In this paper, we discuss practical approaches that exploit existing features of BitTorrent: tPEX for the traffic localization and gTFT for the redirection of the inter-domain traffic. Through simulations, we examine the feasibility of tPEX, gTFT, tPEX+gTFT, and Delay as the practical win-win approach by examining the traffic volume, the charging volume, and the download performance. tPEX is good to reduce the inter-AS traffic volume, the charging volume, and the download completion time. gTFT can redirect some portion of the inter-AS traffic from the transit links to the peering links, but it loses its ability to redirect the inter-AS traffic when tPEX is applied. Even though Delay is good to reduce the inter-AS traffic volume and the charging volume, it significantly increases the download completion time. Therefore, we can say that tPEX can be the practical win-win approach. We also briefly show that the application of tPEX or gTFT to target AS affects the performance of its neighboring AS, which leads us to study the effect on neighboring AS in detail as future work. We also plan to study the relationship between the performance improvement by tPEX and various factors including the number of local peers, the aggregate upload capacity, and the number of download limits.

References

[1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of P2PEcon*, 2003.

[2] D. Wu, P. Dhungel, X. Hei, C. Zhang, and K. W. Ross, "Understanding peer exchange in bittorrent systems," in *Proc. of IEEE P2P*, 2010.

[3] F5 White Paper, "Bandwidth management for peer-to-peer applications," http://www.f5.com/pdf/white-papers/rateshaping-wp.pdf/.

[4] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Portal for (P2P) applications," in *Proc. of ACM SIGCOMM*, 2008.

[5] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P users cooperate for improved performance?," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, 2007.

[6] D. Saucez, B. Connet, and O. Bonaventure, "Implementation and preliminary evaluation of an isp-driven informed peer selection," in *Proc. ACM CoNEXT*, 2007.

[7] J. Seedorf, S. Niccolini, M. Stiemerling, E. Ferranti, and R. Winter, "Quantifying operational cost-savings through alto-guidance for p2p live streaming," in *Proc. of ETM*, 2010.

[8] F. Lehrieder, S. Oechsner, To. Hobfeld, D. Staehle, Z. Despotovic, and W. Kellerer, "Mitigating unfairness in locality-aware peer-to-peer networks," *International Journal of Network Management*, vol. 21, issue 1, pp.3-20, Jan. 2011.

[9] H. Lee, A. Nakao, and J. Kim, "BiCo: Network operator-friendly p2p traffic control through bilateral cooperation with peers," *Elsevier Computer Networks*, vol. 55, issue 9, pp.2023-2034, June 2011.

[10] ALTO (Application-Layer Traffic Optimization), http://datatracker.ietf.org/wg/alto/charter/.

[11] D. Choffnes and F. Bustamante, "Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems," in *Proc. of ACM SIGCOMM*, 2008.

[12] C. H. Hsu and M. Hefeeda, "Isp-friendly peer matching without ISP collaboration," in *Proc. of ROADS Workshop*, 2008.

[13] S. L. Blond, A. Legout, and W. Dabbous, "Pushing bittorrent locality to the limit," *Elsevier Computer Networks*, vol.55, no.3, pp.541-557, 2011.

[14] C. Zhang, P. Dunghel, D. Wu, and K. Ross, "Unraveling the bittorrent ecosystem," *IEEE Trans. on Parallel and Distributed Systems*, 2010.

[15] M. Yoshida and A. Nakao, "A resource-efficient method for crawling swarm information in multiple bittorrent networks," in *Proc. of International Symposium on Autonomous Decentralized Systems*, 2011.

[16] M. Yoshida and A. Nakao, "Measuring bittorrent swarms beyond reach," in *Proc. of IEEE P2P*, 2011.

[17] 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.

[18] M. Piatek, H. V. Madhyastha, J. P. John, A. Krishnamurthy, and T. Anderson, "Pitfalls for isp-friendly p2p design," in *Proc. of Hot Topics in Networks Workshop*, 2009.

[19] L. Jun, Z. Shunyi, L. Shidong, and X. Ye, "Active p2p traffic identification technique," in *Proc. of IEEE CIS*, 2007.

[20] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12-20, 2010.

[21] Cisco, Network-based application recognition (NBAR), http://www.cisco.com/en/US/products/ps6616/products_ios_protocol_group_home.html.

[22] SANDVINE Report, Global Internet Phenomena Report, http://www.sandvine.com/downloads/documents/Phenomena_2H_2012/Sandvine_Global_Internet_Phenomena_Report_2H_2012.pdf

[23] SANDVINE, http://www.sandvine.com/.

[24] Kontiki, http://www.kontiki.com/.

[25] Torrage, Torrent storage cache, http://www.torrage.com/.

[26] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/.