On Tracker Selection for Peer-to-Peer Traffic Locality

Haiyang Wang[1] Jiangchuan Liu[1] Bo Chen[2] Ke Xu[3] Zhen Ma[3]
[1]{hwa17, jcliu}@cs.sfu.ca
School of Computing Science, Simon Fraser University, British Columbia, Canada
[2]bochen@cs.ubc.ca
School of Computing Science, University of British Columbia, British Columbia, Canada
[3]{xuke, mazhen}@csnet1.cs.tsinghua.edu.cn
Department of Computer Science and Technology, Tsinghua University, Beijing,China

***Abstract*—BitTorrent (BT) is an extremely successful peer-to-peer (P2P) application providing efficient file sharing over the Internet. The ever-increasing traffic among the peers has also put unprecedented pressure to Internet Service Providers (ISPs). P2P locality has therefore been widely suggested, which explores finding local resources to optimize the cross-ISP/AS traffic. However, the ISPs would fail to reduce the cross-AS traffic if they could not control the neighbor selection of their P2P subscribers.**

In this paper, we examine the applicability of P2P locality through real-world measurement. We find that the widely deployed load balance trackers will greatly reduce the efficiency of traffic locality. Due to peers' random tracker selection, there is no grantee that the peers will always choose the modified trackers as we expected.

Fortunately, our investigation of the AS-Tracker relationship indicates that if we carefully select the trackers during the locality deployment, most peers can still be controlled by the ISPs with relatively high probability. A machine learning based model is then proposed to quantify the similarity of trackers' peer distribution. Our trace-based simulation shows that, the similarity value can provide useful hints to enhance P2P locality. In particular, the peers are more likely to be optimized with higher probability. Moreover, the learning of tracker similarity does not require the global knowledge of Internet trackers, which can hardly be obtained by the individual ISPs.

I. Introduction

In recent years, BitTorrent (BT)-based systems have become extremely popular for content sharing over the Internet, accounting for over 35% of the network traffic [1]. This is even more than the total traffic of all other peer-to-peer applications, e.g., Gnutella [2], KaZaa [3], eDonkey/eMule [4], and etc. Its exceptional scalability and robustness come from the enormous computation, storage, and communication resources collectively available at participating peers. However, the ever-increasing traffic among the peers has also put unprecedented pressure on Internet Service Providers (ISPs). In particular, even though many BT peers interested in identical contents are located in the same or nearby Autonomous Systems (ASes), they are unnecessarily connected through remote peers in the existing BT systems, thereby persistently increasing the costly cross-AS traffic.

In order to alleviate the cross-AS traffic, P2P locality [5] has been widely suggested. Different with caching or blocking the P2P traffic, this method explores the access of existing localities to reduce the long-haul traffic. However, the ISPs have to control the neighbor selection of most peers and let them know the local peers[1]. Otherwise, the ISPs would fail to reduce the cross-AS traffic.

How to control more peers is therefore a critical problem during the locality deployment. The modification of Internet trackers is seemingly an efficient approach yet it has many problems. For example, how many trackers should be modified for a given ISP? Which trackers should be modified with higher priority? These problems remain largely unclear for the real-world deployment of P2P locality.

In this paper, we for the first time investigate the tracker selection problem of P2P locality. Based on real-world measurement, we find that the widely deployed load balance trackers will greatly reduce the efficiency of P2P locality. Due to peers' random tracker selection, there is no grantee that the peers will always choose the modified trackers as we expected. Our experiment shows that the prioritized selection of the popular trackers can achieve only very limited performance. In particular, if the torrents can only cite one tracker in their metainfo files, the Top-10 most popular trackers can optimize 50% peers for sure; however, these trackers will only optimize 7% peers under the realistic multiple-tracker environment.

How to control a large percentage of peers is indeed a challenging job for the ISPs. Therefore, we investigate the peers' geographical distribution across Internet trackers and observe a very strong correlation (0.7046) between the ASes' peer popularity and the ASes' tracker popularity[2]. Moreover, we find that the peer distribution on Internet trackers exhibits strong geographic locality that could be explored. We believe that for a given ISP, if we carefully deploy the P2P locality on

[0]Jiangchuan Liu and Haiyang Wang's work is supported by a Canada NSERC Strategic Grant, an NSERC Discovery Grant, an NSERC DAS grant, and an MITACS Project Grant; Ke Xu and Zhen Ma's work is supported by a 973 Project of China (2009CB320501).

[1]It is worth noting that an important issue is the existence of local peers; our study over this problem is discussed in [6]

[2]The number of trackers that manage the peers of a given AS.

a selective set of trackers, most peers can be controlled with high probability.

During the investigation of tracker selection, we find that some trackers have very similar peer distribution (discussed in Section V, Figure 5) that should be clustered and modified at the same time. A machine learning based model is then proposed to quantify the similarity of trackers' peer distribution (we call this metric *tracker similarity*). Our trace-based simulation indicates that the learning of tracker similarity can enhance the control of more peers and provide useful hints during the locality deployment.

The rest of this paper is organized as follows. In Section II, we illustrate the related work. Section III and IV presents our measurement foundations and problem statements respectively. Section V investigates the AS-Tracker relationship based on the real-world data. The machine learning model is discussed and validated in Section VI. The tracker deployment method is then evaluated by our trace-based simulation in Section VII. Finally, Section VIII concludes the paper.

II. Related Work

The heavy and long-haul traffic of P2P applications raises a significant challenge to the ISPs. Therefore, P2P locality has attracted an increasing amount of attention in recent years following the pioneer work of T. Karagiannis et al. [5]. In order to further quantify its benefits, S. L. Blond et al. [7] showed under a controlled environment that high locality values (defined by [5]) yield up to two orders of magnitude savings on cross-AS traffic, without any significant impact to the peers' download completion time. Since the basic locality mechanism is proven efficient, many novel locality methods have been proposed; for example, H. Xie et al. [8] suggested cooperation between peer-to-peer applications and ISPs by a new locality architecture, namely, P4P, which can reduce both the external traffic and the average downloading time. D. R. Choffnes et al. [9] proposed Ono, a BitTorrent extension that leverages a CDN infrastructure, which effectively locates peers that are close to each other.

For the implementation of traffic locality, most existing studies are based on the tracker-side investigations. R. Bindal et al. [10] examined a novel approach to enhance BitTorrent traffic locality, namely, *biased neighbor selection*. Using this method, a peer chooses the majority, but not all, of its neighbors from peers within the same ISP. B. Liu et al. [11] evaluated various locality implementations and shown that the *tracker locality* (such as biased neighbor selection) will sometimes uneven the downloading time among peers; yet, this policy can indeed achieve the shortest *AS hop count* (see definition in [10]) among all the implementations. A recent study from M. Piatek et al. [2] shown that a "win-win" outcome is unlikely to obtain for the ISPs during the locality; the reason is that reducing inter-domain traffic reduces costs for some ISPs, while it also reduces revenue for others. R. Cuevas et al. [12] also investigated the maximum transit traffic reduction as well as the "win-win" boundaries across the ISPs.

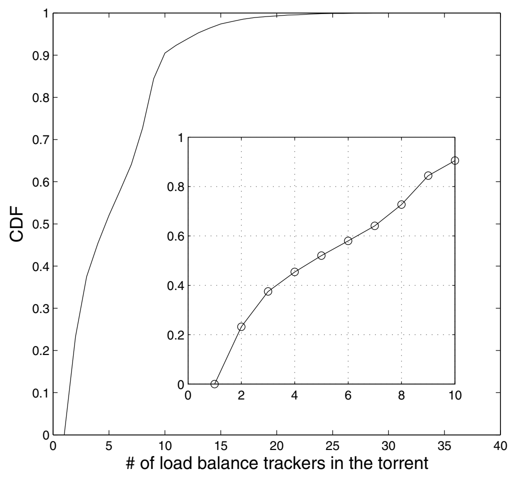


Fig. 1. Popularity of load balance trackers

Considering the measurement of BitTorrent system, most existing studies are focusing on the swarm evolution [13], peers downloading speed [14], choke algorithm and the content pieces selection algorithm [13][15]. For example, a recent study from C. Zhang et al. [16] examined BitTorrent darknets from macroscopic, medium-scopic and microscopic perspectives and investigated the properties of private BitTorrent sites.

However, the deployment issue of P2P locality is seldom discussed in the existing studies. Our work reveals the challenge of controlling the peer for the ISPs based on real-world measurement. The relationship between Internet ISPs and trackers is also explored for the first time to guide the locality deployment.

III. Measurement Foundations

A. Methodology

We first extracted a large collection of real torrents as advertised by www.btmon.com, one of the most popular torrent sites, from August 2008 to July 2009. We developed a script to automatically detect the 'href' field in each given HTML file and download the metainfo files ending with '.torrent'. In order to balance accuracy and measurement overhead, we randomly selected $8,806$ torrents in our study.

We then ran a modified version of CTorrent (a typical BitTorrent client in FreeBSD) [17] on the PlanetLab nodes. Different from conventional pure PlanetLab experiments in which the clients communicate with others within the PlanetLab only, our modified CTorrent clients actively joined existing torrents in the global Internet and recorded the observable peer information obtained from the trackers and from other peers over time. As such, the small set of controlled PlanetLab nodes were able to capture the information of most peers in the torrents, in particular, their IP addresses. With a maximum of 50 initial peers from the trackers, we successfully detected

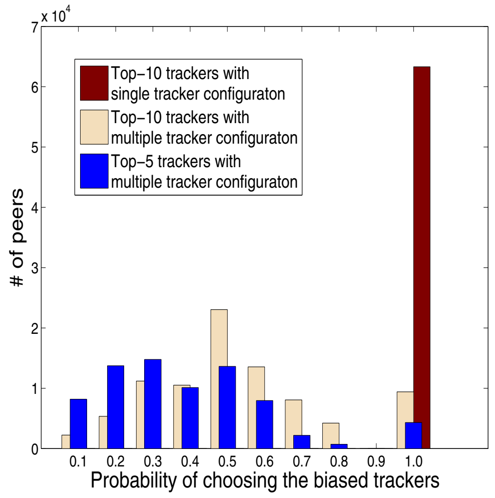


Fig. 2. Efficiency of tracker popularity based deployment

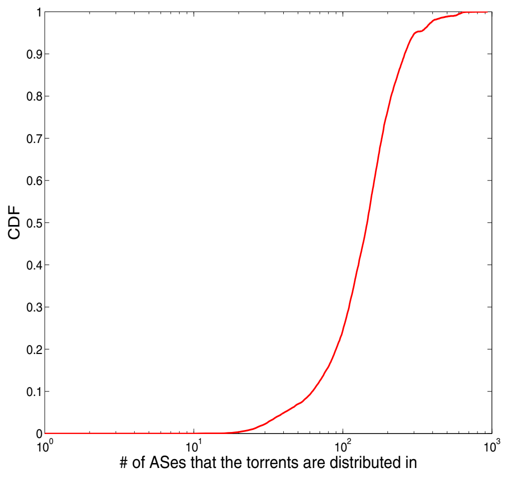


Fig. 3. Peers' location in the torrents

the IP addresses of over 95% peers[3] for most of the torrents.

Except for retrieving the peer existence and address information, our PlanetLab clients do not download or upload any real data of the shared contents. Hence, no copyrights have been violated, and the impact to the PlanetLab traffic and to the operations of normal trackers/peers were minimized. The scanning efficiency of the experiment is also very high, with most of the torrent being finished within a short timeframe (< 20 seconds); in other words, the detected peers can be considered as being online simultaneously, which is important to our discussions in the next section. To avoid biases, we have also filtered out all the PlanetLab nodes in the data presented in the following analysis.

[3]This ratio is calculated by comparing the number of detected peers with the total number of peers as advertised by the tracker of a torrent.

B. Two Basic Relationships

We use $\aleph$ to denote all the ASes on the Internet, $\Re$ to denote the set of existing torrents and use $\Im$ to denote the set of Internet trackers. We define three random variables A, S and T; A is a random variable that takes on different ASes $a \in \aleph$; S takes on values over the set of Internet torrents $\Re$,and T takes on values over the set of trackers $\Im$ that managing the torrents.

Based on the above components, two relationships can be learnt from the measurement: (1) The relationship between S (torrents) and T (trackers); (2) The relationship between A (ASes) and S (torrents). We use binary matrix R_1 to define the relationship (an un-normalized frequency table) between S and T, and this matrix is learnt directly from the metainfo (.torrent) files; each element of $R_1(s,t)$ is of a binary value, indicating whether torrent s includes tracker t in its metainfo file for load balance propose (1-Yes, 0-No). Note that we are particularly focusing on the load balance trackers in this paper because this configuration will potentially affect the deployment of locality (the details will be discussed later in the next section).

On the other hand, we use matrix R_2 to refer to the frequency table of S and A. This matrix is learnt from the IP/AS information of the probed BT peers; each element $R_2(s,a)$ is an integer which refers to the number of peers (in torrent s) that belong to AS a. The peers' AS information is learnt by the 'whois' command on the Linux system, and most replies are from 'whois.cymru.com'.

IV. Challenge of Controlling The Peers

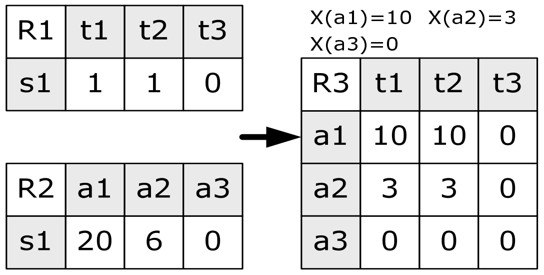
For traffic locality, the optimization of peers' neighbor selection is an attractive idea. However, the ISPs would fail to reduce the cross-AS traffic if they lose the control of most peers. In this section, we will explore the two based relationships and discuss the challenge of controlling the peers for traffic locality.

A. Popularity of Load-balance Trackers

It is troublesome to control the numerous Internet peers on an individual basis, and it would be more efficient to control them via tracker modification instead. In order to clarify the real-world efficiency of this approach, we first explore the relationship between Internet trackers and torrents.

It is well known that the latest BitTorrent metainfo file can include multiple tracker sites stored in the *announce-list* section [18]. This multi-tracker configuration allows peers to connect to more than one tracker, which brings two tangible benefits: (1) better accommodates tracker failures; (2) balances loads among the trackers. In other words, some trackers are now managing the same set of torrents cooperatively and help increasing the availability of Internet torrents.

Unfortunately, we find that this extension will reduce the efficiency of controlling the peers. It is well known that the BitTorrent protocol only allows a peer to associate with one tracker per content to reduce the possible overheads; therefore, unless we can modify all the trackers in every torrents'



R1	t1	t2	t3
s1	1	1	0

R2	a1	a2	a3
s1	20	6	0

X(a1)=10 X(a2)=3 X(a3)=0

R3	t1	t2	t3
a1	10	10	0
a2	3	3	0
a3	0	0	0

Fig. 4. An example of computing R_3

announce-list, it is hard to grantee that the peers will always connect to the modified biased trackers as we expected.

This is seemingly an extreme case because some Internet torrents may not include any load balance trackers. In order to investigate the generality of such a configuration, we examine the $8,806$ torrents in our dataset. We record the announce-list of each torrent, pick out the load balance trackers and then compute the number of trackers that have been used by the torrents. Figure 1 indicates that more than 80% torrents have specified at least two trackers for load balance purpose, and a few torrents even have announce-lists of several hundred trackers. This is much higher than an earlier measurement in 2007 [19] (observed multi-trackers in 35% of the torrents), and thus suggests that the load balance configuration has been quickly recognized and deployed in the BitTorrent community. Note that the possible tracker failure as well as their long-term availability is beyond the scope of this paper. Due to the lack of this highly dynamic information, we temporarily ignore the backup trackers during the discussion and reserve this issue for our future studies.

In order to clarify the impact of the load-balance trackers, we explore the peer control problem in AS#3352 as an example (this is the most popular AS with $126,133$ peers in our measurement). The peers in this AS are distributed in $6,065$ torrents that managed by 384 Internet trackers.

Figure 2 presents the results of how many peers will be controlled by the P2P locality with what kind of probability. We first simulate the single tracker based configuration by randomly removing some load-balance trackers in torrents' announce lists. We can see that if the torrents can only cite one tracker in their metainfo files, $62,817$ peers in AS#3352 will be controlled for sure if we deploy P2P locality on the Top-10 most popular trackers. Although other peers will have no chance to be optimized, 50% peer coverage is still encouraging especially considering the overheads of modifying only 10 trackers.

However, in the real-world multiple tracker environment, the Top-10 most popular trackers can only control 7% peers ($9,414$ out of $126,133$) for sure, and leaving 30% ($38,505$) peers that have no chance to be optimized (with the probability of 0). We also investigated the efficiency with the prioritized selection of more than 10 trackers. Unfortunately, the performance is remain limited; in general, more than 150 trackers are required to cover less than 50% peers for the ISPs.

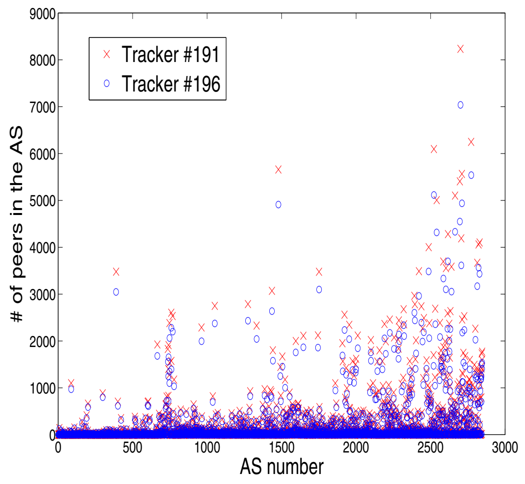


Fig. 5. Two very similar trackers in our measurement

V. Peer Distribution on Internet Trackers

In this section, we will clarify two questions: First, how many trackers are required to enforce the peer control for the ISPs; second, which trackers should be modified with higher priority? During the discussion, we believe that the ISPs are more interested in the optimization of their own P2P subscribers. As shown in Figure 3 (learnt from matrix R_2), the peers of most torrents are distributed in more than 150 ASes; therefore the ISPs generally have no incentive to control all the peers in the torrents.

A. Relationship Between Internet ISPs and Trackers

In this part, we for the first time trying to clarify the relationship between the ISPs and trackers based on real-world measurement. The main approach is to compute such a relationship based on the existing relationships R_1 (torrents-trackers) and R_2 (ASes-torrents). We assume that all the peers will randomly select one tracker among multiple load balance trackers with no special preference. Note that this approximation is based on the BitTorrent protocol [20] and its multi-torrent specification [18] which will not bias our investigation.

We prefer to compute such a relationship because the exact AS-Tracker relationship is highly dynamic overtime. The peers could randomly switch to other trackers after any short time failure or task break (stop and restart of downloading). For example, even an real-world snap shot shows peer x is managed by tracker y, we cannot expect such a relationship in the future time slots. Therefore, the real-time snap shots can hardly be used for optimization. More importantly, compare to the two basic relationships (R_1 and R_2), this highly dynamic information is also hard to obtain by the ISPs.

We will now present the details of computing the AS-Tracker relationship (we use matrix R_3 to refer this relationship). Each component of $R_3(t, a)$ indicating how many peers that located in AS a are managed by tracker t. Figure 4

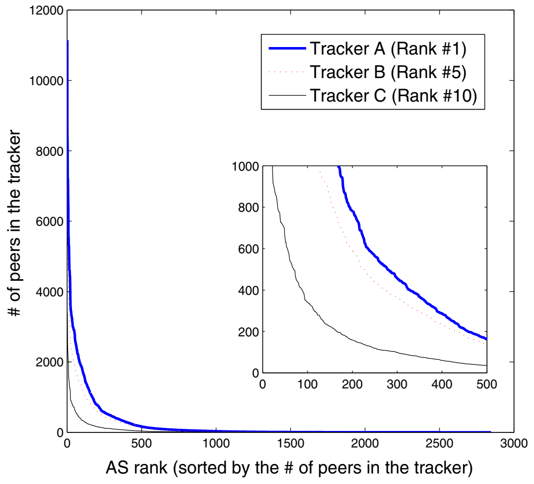


Fig. 6. Distribution of peers' location on three very popular Internet trackers (all sorted in descend order)

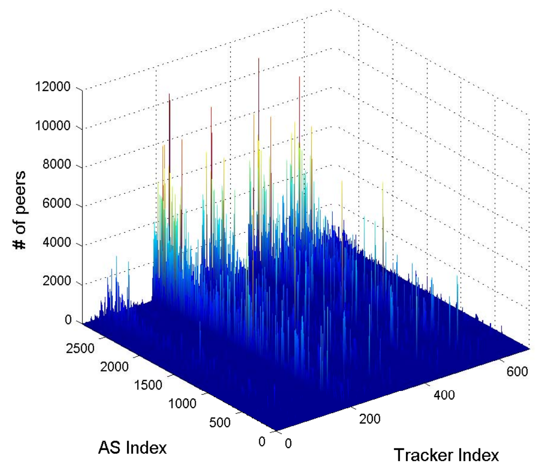


Fig. 8. The big picture of R_3

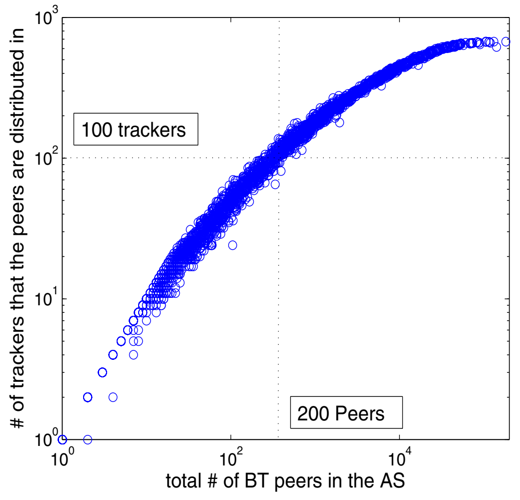


Fig. 7. Correlation of AS's peer population and the number of trackers that the peers are distributed in (each circle in the figure refers to an Internet AS)

presents an example on how to compute R_3 when matrix R_1 and R_2 only contain the information of one torrent (s_1). As shown in Figure 4, we first electing the elements in the row of R_2. For each element, we divide it by the sum of the corresponding row (refers to a torrent) in the binary matrix R_1 (this sum indicates how many load balance trackers are managing this torrent) and then get a real value x as follows:

$$x = R_2(s_1, a_1) / \sum_{t \in T} R_1(s_1, t) \tag{1}$$

After that, each column of $R_3(T, a)$ (for example, the first column a_1) can be learnt by processing a dot product between the corresponding row in R_1 and x:

$$R_3(T, a_1) = (R_1(s_1, T) \cdot \lceil x \rceil)^{T_{transpose}} \tag{2}$$

The remaining columns of R_3 can be computed respectively based on the above method. Moreover, if more torrents are available, their results can be accumulated together and get the final matrix of R_3.

B. Discussion

The big picture of matrix R_3 (learnt from our dataset that includes 2846 ASes and 683 trackers) is shown in Figure 8. Based on this relationship, we first discuss the overhead of locality deployment, that is, how many trackers are required to control the peers for the ISPs?

As shown in Figure 7, we observe a very strong correlation (0.7046) between the two data sets (total number of peers in the AS and the trackers that managing the peers for this AS). An intuitive explanation is that the ISPs are able to predict how many trackers are required by monitoring the population of their P2P subscribers. It is worth noting that this overhead is relatively high for small ISPs; in particular, 100 trackers are required to control only 200 peers for some small ISPs. On the other hand, less than 300 trackers can control more than $15,000$ peers in those large ISPs. Note that in this case, the peers will be controlled with the probability of 1.0.

The follow-up investigation shows that this overhead can be reduced if we only want to optimize most peers for the ISPs. In Figure 6, we can see that the peer distribution on Internet trackers can roughly be fitted by exponential distributions which exhibit strong geographical locality. Moreover, we find that some Internet trackers have very similar peer distribution. For example, as shown in Figure 5, we can see that tracker#191 and #196 are managing similar number of peers over the Internet ASes. Naturally, if the tracker #191 is modified to control the peers for a given ISP, tracker #196 should also be modified at the same time. We believe that these similar trackers should be clustered and modified together to enforce the peer control. However, different from tracker popularity, how to quantify such a similarity is a challenging problem.

VI. Quantify The Similarity Among BitTorrent Trackers

In this section, we will quantify the similarity of peer distribution on different trackers. We believe that this metric can provide more information than that of tracker popularity; therefore, some useful hints could be gained to guide the deployment of locality. Note that simply compute their value difference is not a good choice because the unpopular trackers will more likely to be clustered as 'similar trackers'.

A. Modeling of Tracker Similarity Based on Peers' Location

Recently, Salakhutdinov and Hinton [21] propose the use of a restricted boltzmann machine to learn document similarities. A closely related approach using deep auto-encoder [22] is also shown to perform well in learning representations where documents of different categories can be clearly separated. It is worth noting that the learning process is purely unsupervised, i.e. the true category information of the documents is not used to learn the representations, or equivalently speaking, the process relies on and learns to reveal the intrinsic categorical similarities of the documents. Motivated by the success in document retrieval, we seek to learn representations for the trackers such that the similarity between their peer distributions, though unintuitive to work with in their raw form, can be well characterized by a simple distance metric in the learnt representation.

For simplicity, we first preprocess the relation matrix R_3 into a data set D of binary tracker profiles. The preprocess steps include performing standardization over the columns and then the rows of R_3, and thresholding the resultant matrix at 0. After preprocessing, D contains profiles of all T trackers ($T = 683$). D_t, the t-th row of D, is a d-dimensional binary profile for tracker t, with $d = 2846$.

Then we use a one-layer auto-encoder to learn a low-dimensional, binary representation from the data. Our preference over this simplistic model is justified by the fact that complex models will suffer from *over-fitting* on small data sets[4]. The idea of an auto-encoder is to extract common patterns (combinations of ASes) in the tracker profiles and reconstruct the profiles by selectively and additively combining the patterns. The representation for a specific tracker profile is such that each dimension indicates the presence/absence of a pattern in the profile. This representation is then used as the binary coefficients for linearly combining the patterns to reconstruct the profile. An alternative view would be that each dimension of the representation forms a binary clustering of the profile. When all the dimensions are combined, it implicitly creates a cluster model where the number of clusters is exponential in the number of dimensions. Finally, the whole representation can be interpreted as the binary encoding of the index to this cluster model.

[4]One intuitive explanation of over-fitting is that the number of independent parameters in complex models is so large that small data sets fail to impose enough constraints to train them. In our case, D only contains 683 profiles, which is small with respect to the dimensionality (2846) of the data.

More specifically, let $H_t \in \mathbb{R}^{N_h}$ be the representation of D_t, where N_h is the dimensionality of the representation. The auto-encoder is parameterized by $\theta = \{W \in \mathbb{R}^{d\times N_h}, b \in \mathbb{R}^d, c \in \mathbb{R}^{N_h}\}$. The probability of the representation H_t and the reconstruction $\hat{D}_t$ are given by:

$$P(H_{tj} = 1|D_t, \theta) \equiv \sigma\left(b_j + \sum_{i=1}^{d} W_{ij} D_{ti}\right), \forall j = 1, 2, \ldots N_h$$

$$P(\hat{D}_{ti} = 1|H_t, \theta) \equiv \sigma\left(c_i + \sum_{j=1}^{N_h} W_{ij} H_{tj}\right), \forall i = 1, 2, \ldots d$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function.

To compute the reconstruction given a tracker profile, we have to integrate out all the possible representations, which is computationally prohibitive. Fortunately, a mean-field approximation typically works reasonably well. Reconstruction with mean-field approximation is given by:

$$P(\hat{D}_{ti} = 1|D_t, \theta) \approx \sigma\left(c_i + \sum_{j=1}^{N_h} W_{ij} P(H_{tj} = 1|D_t, \theta)\right) \quad (3)$$

Finally, we can sample from $P(\hat{D}_{ti} = 1|D_t, \theta)$ to obtain the binary reconstruction, if necessary.

The auto-encoder is trained by minimizing the square reconstruction error between the tracker profile and the corresponding reconstruction over a training set of T tracker profiles:

$$E(\theta) \equiv \frac{1}{2T} \sum_{t=1}^{T} \sum_{i=1}^{d} (P(\hat{D}_{ti} = 1|D_t, \theta) - D_{ti})^2$$

Since N_h is typically smaller than d, it provides an information bottleneck which prevents the auto-encoder from learning trivial patterns such as the singletons. However, if N_h is too small, information in the representation is severely limited and sacrifices reconstruction performance. On the other extreme, setting N_h too large leads to over-fitting on our dataset, making it difficult to generalize to unseen tracker profiles. The concerns above call for proper model regularization and selection.

To encode our preference for the simplest competent model, we apply L_2 regularization on the parameters and use cross-validation for selecting the most appropriate N_h. The regularized objective is given by:

$$E_s(\theta) = E(\theta) + \frac{\alpha}{2} \sum_{i=1}^{d} \sum_{j=1}^{N_h} W_{ij}^2$$

where α is a small constant that trades off our preference for simplicity and accuracy.

B. Training Protocol

We perform cross-validation to select the best hyper-parameter settings for α from the set of $\{10^{-4}, 10^{-2}, 1\}$

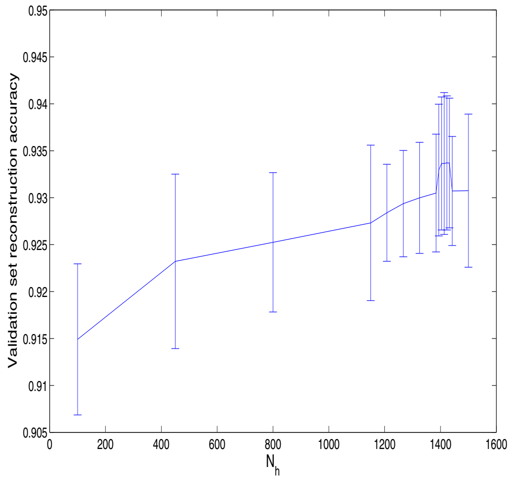


Fig. 9. The growth of validation set reconstruction accuracy as a function of N_h (dimensionality of the representation) with a weight decay of $\alpha = 0.0001$. The bars show one standard deviation above and below the mean accuracy. Since a $5-$nary search is performed, data will be denser at regions with high mean accuracy.

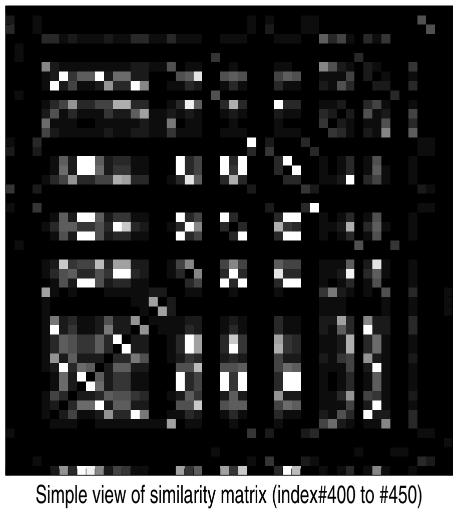


Fig. 10. A simple view of similarity matrix

and N_h from the interval $[20, 1500]$ of integers[5]. 5 trials are performed for each parameter configuration. For each tiral, we first randomly split D into a training set of 600 profiles and a validation set of 83 profiles. We then train on the training set by running a maximum of 200 iterations of L-BFGS [23] using Mark Schmidt's minFunc [24], and compute $E_s(\theta)$ on the validation set. The parameter configuration that yields the best reconstruction performance in the validation set, averaged over 5 trials, is selected. Finally, we compute the inverse of the Hamming distance in the representation between trackers as their similarity measure for deployment.

The model's sensitivity to the choices of N_h between $[20, 1500]$ is shown in Figure 9. The overall performance is rather stable and preferred in the range of $[1000, 1500]$. Considering the size of our data set, we do not believe that the data, in a statistically significant matter, favors one choice of N_h over the other within this range. Therefore we simply use the final value $N_h = 1423$ returned by the 5-nary search.

A simple view of the computed similarity matrix is shown in Figure 10, where the intensity of the squares indicates the similarity value among the trackers. We can see that there are some noticeable small clusters, which generally include less than 10 trackers in this matrix. This observation confirms our measurements that some trackers are indeed working together and managing the similar sets of peers. Note that the random failure of Internet trackers will potentially affect the computing of R_3 as well as the trackers' similarity values. However, the existing studies on tracker availability shows that most Internet trackers have quite good availability (for example, Figure. 3 in [18]). To this end, we temporarily ignore the availability of different trackers to reduce the complexity of our model. We will aim to combine some time-related parameters and further enhance the model in our future investigations.

[5]For efficiency concerns, we assume monotonicity in validation accuracy and use 5-nary search to find N_h in the interval. To optimize a function F in an interval, the 5-nary search evaluates F at 5 evenly-spaced samples of the current interval, then shrinks the length of the interval by a factor of 5 while re-centering it at the sample with the minimum F value. This procedure recurses until the interval becomes unity.

VII. Application and Evaluation

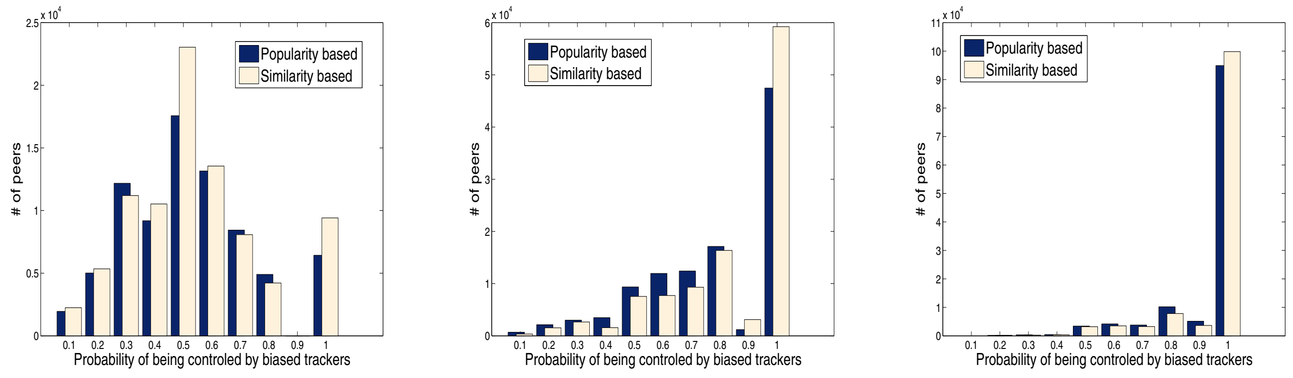
In this section, we will discuss the possible applications of similarity value as well as its performance for the locality deployment. During the evaluation, we are focusing on the control of peers, that is, how many peers will be controlled by the P2P locality (modified biased trackers) with what kind of probability.

A. *Similarity Based Tracker Deployment*

The design of our similarity based tracker deployment is as follows: Assuming that the ISP can be aware of the most popular tracker (that managing its own peers) by traffic monitoring (for example, tracker X is the most popular one). The ISP will first select this tracker and then the most similar trackers with tracker X one by one based on the computed similarity matrix. This approach is intuitive, since tracker X is managing a lot of peers for this ISP, select the most similar trackers with tracker X can enforce the control of more peers. Note that we are discussing the real-world deployment of P2P locality; the proposed method has to be simple and easy to carry out.

We first evaluate this method via a small ISP with $141, 154$ peers (these peers are all located in one AS). Our simulation is based on real-world measurement of R_1 and R_2. The simulator will calculate the possible tracker candidates for the peers and compute the probability that the peers will be controlled; the input of this simulator is a list of modified trackers.

As shown in Figure 11, we can see that the similarity based deployment can indeed control more peers with higher



(a) (b) (c)

Fig. 11. Peers that will choose the biased trackers with certain probability (with the total number of $141,154$ peers): (a) 20 trackers are modified for locality purpose; (b) 50 trackers are modified for locality purpose; (c) 160 trackers are modified for locality purpose;

TABLE I
NUMBER OF PEERS THAT WILL CHOOSE THE MODIFIED BIASED TRACKERS IN CERTAIN PROBABILITY (FOR FIGURE.12)

	20%	**40%**	**60%**	**80%**	**100%**
S(20)	5343	10525	13553	4218	9414
P(20)	5013	9183	13156	4900	6428
S(50)	1524	1558	7711	16354	59238
P(50)	2114	3476	11946	17117	47469
S(160)	170	440	4167	10193	94930
P(160)	178	344	3485	7817	99846

probability. For example, in Figure 11(b), when 50 trackers can be modified, $47,496$ peers will be controlled for sure by the popularity based deployment whereas $59,238$ peers will be optimized for sure by the similarity based deployment. The more detailed data of Figure 11 can be found in Table I, where $S(n)$ and $P(n)$ refer to the popularity based and similarity based approach respectively; n refers to the number of trackers that can be modified for P2P locality.

It is worth noting that in Figure 11(c), when 160 trackers can be modified, the improvement of similarity based approach is decreased. In order to understand this feature, we further investigate a larger ISP that combined by multiple ASes. In this case, the ISP includes $689,453$ peers that managed by 683 trackers. We modify all the trackers one by one with both sequences that computed by the similarity and popularity based approaches respectively. Figure 12(a) presents how many peers will be optimized by the locality trackers for sure (with the probability of 1.0). We can see that the improvement of similarity based approach keeps increasing at the beginning and soon reaches its maximal when x is around 100. Moreover, if we cumulate the peers that will be controlled with the probability between $[0.8, 1.0)$ in Figure 12(b), we can see that more than $100,000$ peers will be optimized with higher probability.

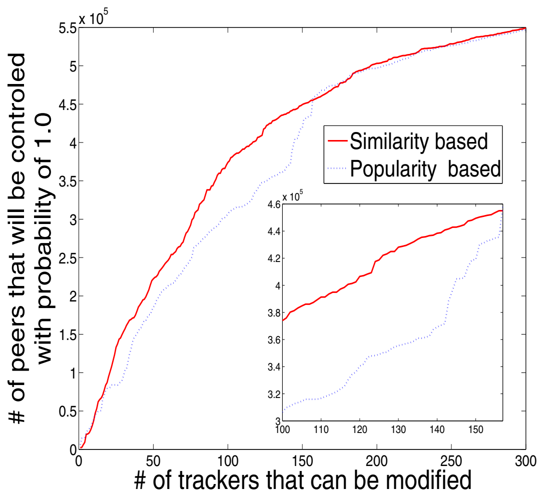
On the other hand, we again find that the performance of the two approaches will become nearly identical after $x = 157$. In fact, we observe similar results in other ASes where the disparity of the two approaches becomes very small when x is around 160. A closer look indicates that 160 trackers can indeed cover more than 75% peers for sure in most ISPs. An intuitive explanation is that, when a near-complete deployment (a great number of trackers can be modified at the same time) is achievable, the sequence of tracker selection is no longer that important. However, it is hard to modify 160 trackers at the same time in the real-world deployment, especially considering the existence of non-cooperative trackers.

B. Discussion

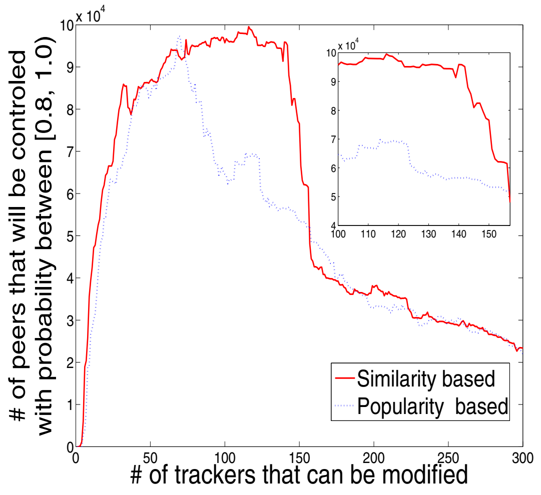
Our investigation demonstrates the usefulness of trackers' similarity value. It is worth noting that we are trying to learn the rules during the locality deployment but not aiming to find an optimal deployment algorithm. In fact, we have mentioned that the relationship between peers and trackers is highly dynamic over time. The ISPs cannot detect this information from time to time and keep adjusting their tracker deployment. Due to these reasons, a global optimal algorithm can hardly be applied in the real-world deployment.

What we want to prove is that the similarity value can provide useful hints to improve P2P locality (in particular, help the ISPs to control more peers). Based on this metric, some simple and easy-to-deploy methods can be designed to achieve pretty good efficiency. It is also worth noting that some existing studies suggest to organize Internet trackers together to improve BitTorrent's downloading performance [25]. The similarity value also has the potential to guide the clustering as well as the organization of a tracker overlay that proposed by [25].

In this paper, we are trying to address the key issues of deploying the P2P locality; some detailed problems are therefore ignored during the discussion. First, the time consistency of similarity value. In other words, will the similarity value change rapidly for the trackers? Recall that the similarity value is computed based on R_1 and R_2 that are both stable over time. We have also briefly compared the learnt manifold as well as the similarity values with other snap-shots in our measurement



(a)



(b)

Fig. 12. More trackers can be modified for locality purpose: (a) Number of peers that will choose the biased trackers for sure; (b) Number of peers that will choose the biased trackers with probability between [0.8, 1.0).

data. Although some minor diversities are observed, we believe that it can hardly affect the locality deployment.

Second, given the probabilities and the number of peers that can be controlled by the ISPs, can we quantify the saved cross-AS traffic? The modeling of this problem is indeed a challenging job especially considering multiple torrents and the relationship between the ISPs (note that the existing studies are mostly focusing on the traffic saving when the users participate in one torrent). Although the discussion of this model is out the scope of this paper. We are aiming to provide a more extensive discussion in our future studies.

VIII. Conclusion and Future Works

Locality has been widely suggested as a solution to mitigate the cross-AS traffic for the ISPs. Yet, its performance will be greatly reduced if the ISPs cannot control most of their peers.

TABLE II
Top-10 Most Popular Trackers

Rank	Peers	Torrents*	Tracker Sites (URLs)
1	607987	19915	open.tracker.thepiratebay.org
2	593205	16724	trackeri.rarbg.com
3	560580	23386	denis.stalker.h3q.com
4	509140	15308	tpb.tracker.thepiratebay.org
5	504173	12117	vtv.tracker.thepiratebay.org
6	442708	12821	vip.tracker.thepiratebay.org
7	414095	10019	eztv.tracker.prq.to
8	262991	6079	tracker.prq.to
9	184843	3016	tk2.greedland.net
10	142220	3114	www.sumotracker.org

*Note that the torrent level popularity is obtained from the metainfo files which can include multiple trackers.

In this paper, we for the first time investigate this challenge in the real-world deployment. We find that, for most ISPs, the widely used load balance trackers will reduce the efficiency of locality, and the peers will not always select the modified biased trackers as we expected.

In order to address the existing problems, we investigate the peers distribution on Internet trackers. Based on the AS-Tracker relationship, a machine learning model is proposed to quantify the tracker similarity. The evaluation results indicate that the tracker similarity can provide useful hints to guide the locality deployment.

During the investigation, we also find that many Internet trackers, especially some most popular ones, are belonging to public organizations such as Pirate Bay, Demonoid and etc[6]. AS show in Table II, these trackers generally involve in a series of lawsuits, as plaintiffs or as defendants and may hardly cooperate with the ISPs. We find that the performance of P2P locality will be further reduced due to the absence of these trackers.

Considering the the existence of these non-cooperative trackers, we will further discuss the problem of peer control and enhance our model in our future studies.

References

[1] CacheLogic. [Online]. Available: http://www.cachelogic.com/
[2] Gnutella. [Online]. Available: http://www.gnutellaiums.com/
[3] KaZaa. [Online]. Available: http://www.kazaa.com/
[4] Edonkey. [Online]. Available: http://www.edonkey2000.com/
[5] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet Service Providers Fear Peer-Assisted Content Distribution?" in *Proc. ACM/USENIX IMC, 2005*.

[6]After legal conflicts in 2009, both Pirate Bay and Demonoid were taken off-line. However, Demonoid has soon reestablished themselves outside of the USA and may still involve potential copyright problems. On the other hand, Pirate Bay also released an online statement showing that they will be back online shortly.

[6] J. Liu, H. Wang, and K. Xu, "Understanding Peer Distribution in Global Internet," *IEEE Network Magazine, 2010*.

[7] S. L. Blond, A. Legout, and W. Dabbous, "Pushing BitTorrent Locality to the Limit," *INRIA Tech, Rep., 2008*.

[8] H. Xie, R. Y. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: provider portal for applications," in *Proc. ACM SIGCOMM, 2008*.

[9] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," in *Proc. ACM SIGCOMM, 2008*.

[10] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *Proc. IEEE ICDCS, 2006*.

[11] B. Liu, Y. Cui, Y. Lu, and Y. Xue, "Locality-Awareness in BitTorrent-Like P2P Applications," *Proceedings of the IEEE Transactions on Multimedia, 11(3):361-371, 2009*.

[12] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez, "Deep Diving into BitTorrent Locality," *Telefonica Research, Tech, Rep., 2009*.

[13] M. Izal, G. Urvoy-Keller, E. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Life Time," in *Proc. ACM/USENIX IMC, 2004*.

[14] M. Izal, E. B. G. Urvoy-Keller, P. Fellber, A. A. Hamra, and L. Garces-Erice, "Measurement, Analysis and Modeling of BitTorrent-like Systems," in *Proc. The 5th Passive and Active Measurement Workshop, 2004*.

[15] A. Logout, Urvoy-Keller, and P. Michiardi, "Understanding BitTorrent: An Experimental Perspective," *INRIA, Tech, Rep. 00000156, 2005*.

[16] P. Dhungel, D. Wu, Z. Liu, , and K. Ross, "BitTorrent Darknets," in *Proc. IEEE INFOCOM, 2010*.

[17] Ctorrent. [Online]. Available: http://ctorrent.sourceforge.net/

[18] BitTorrent Multi-tracker Specification. [Online]. Available: http://www.bittornado.com/docs/multitracker-spec.txt

[19] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems," in *Proc. IEEE INFOCOM, 2007*.

[20] The BitTorrent Protocol Specification. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html

[21] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, December 2008.

[22] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[23] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of Quasi-Newton Matrices and their use in Limited Memory Methods," *Mathematical Programming, 63, 4, pp. 129-156, 1994*.

[24] M. Schmidt, "minfuc http://people.cs.ubc.ca/ schmidtm/software/minfunc.html," 2005.

[25] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-like Systems," in *Proc. ACM/USENIX IMC, 2005*.