Computer Communications 43 (2014) 64–73

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Strategic bundling for content availability and fast distribution in BitTorrent

Jinyoung Han [a], Taejoong Chung [a], Seungbae Kim [b], Hyun-chul Kim [c,*], Jussi Kangasharju [d], Ted "Taekyoung" Kwon [a,*], Yanghee Choi [a]

[a] School of Computer Science and Engineering, Seoul National University, Seoul 151-742, South Korea
[b] KAIST Institute for Information Technology Convergence, Daejeon 305-701, South Korea
[c] Department of Computer S/W Engineering, Sangmyung University, Cheonan 330-720, South Korea
[d] Department of Computer Science, PL 68 (Gustaf Hallstromin katu 2b) 00014, University of Helsinki, Finland

ARTICLE INFO

Article history:
Received 19 March 2013
Received in revised form 10 January 2014
Accepted 31 January 2014
Available online 8 February 2014

Keywords:
Peer-to-peer
BitTorrent
Content bundling

ABSTRACT

BitTorrent, the immensely successful file swarming system, supports content bundling: a common strategy by which publishers package multiple related files and disseminate them via a single larger swarm. It has been reported that bundling in BitTorrent is wide-spread, currently being done in a subjective and manual manner by individual publishers. This paper is motivated by the following questions: What if bundling is automatically supported by the BitTorrent swarming system (e.g., at a tracker)? By what criteria and how can files be automatically bundled, for better performance of the swarming system? How much performance improvement could be obtained with such an automatic bundling system? To answer the questions, we first suggest nine bundling strategies based on (i) the attributes of the torrents such as title or tags and (ii) time-varying swarm dynamics such as torrent popularity or availability. We then propose and develop a tracker-based bundling system, where all the proposed bundling strategies are implemented and evaluated with a set of real BitTorrent traces. We show that all the proposed bundling strategies outperform no-bundling case in terms of the availability and file download time. The bundling strategy that bundles more popular/available torrents with less popular/available ones outperforms others in most cases. We find that title- and tag-similarity based bundling strategies also provide performance improvement comparable to those of the popularity/availability based bundling strategies.

1. Introduction

According to recent Sandvine's report in 2012, BitTorrent traffic is estimated to account for 10–20% of all the Internet traffic [1]. Although its swarming technique scales well in massive flash crowds for popular files [2,3], BitTorrent often suffers from low availability of unpopular files [4,5]. That is, peers arriving after the initial flash crowd may end up finding the file unavailable [6,4].

Recently, *bundling* in BitTorrent has gained much attention, as it can solve/mitigate the availability problem of unpopular files [4,7] as well as reduce download times [4,7–12]. Here, bundling in BitTorrent is a common strategy by which a publisher packages multiple files into a single torrent and the bundled files are collectively disseminated via a single swarm, instead of via separate swarms. A torrent[1] in BitTorrent contains either a single file or multiple files that can be downloaded collectively in a swarm.

Currently, content bundling in BitTorrent is done by individual publishers, in a subjective, ad hoc, and manual manner; publishers often bundle somehow related files such as multiple episodes of the same movie series (e.g., Shrek 1, 2, and 3) for those who may want to download some or all of them. It has been reported that users choose to download most files (94%) in a bundled torrent [13]. However, in choosing files to bundle, they do not take into account the swarming system performance such as availability and file download time.

This paper is motivated by the following questions: *"What if bundling is automatically supported by the BitTorrent swarming system (e.g., at a tracker)? By what criteria can files be automatically bundled for system performance (e.g., availability improvement or*

* Corresponding authors. Tel.: +82 2 880 9147; fax: +82 2 872 2045. (T.T. Kwon)
E-mail addresses: jyhan@mmlab.snu.ac.kr (J. Han), tjchung@mmlab.snu.ac.kr (T. Chung), sbkim@kaist.ac.kr (S. Kim), hyunchulk@gmail.com (H.-c. Kim), Jussi.Kangasharju@helsinki.fi (J. Kangasharju), tkkwon@snu.ac.kr (T.T. Kwon), yhchoi@snu.ac.kr (Y. Choi).

http://dx.doi.org/10.1016/j.comcom.2014.01.013
0140-3664/

[1] We will use the terms *torrent* and *swarm* interchangeably; the former focuses on files, while the latter focuses on peers.

download time reduction)? How much performance improvement could be obtained with such an automatic bundling system?

As a first step towards developing such an automatic and performance-driven content bundling scheme in the swarming system, we first suggest several practical bundling strategies. The criteria to select which torrents to be bundled together are classified into two categories: (i) similarity among the attributes of the torrents such as titles and tags, and (ii) the time-varying swarm dynamics such as popularity and availability. We then propose and develop a tracker-based bundling system to substantiate the proposed bundling scheme, where the performance of all the proposed bundling strategies are evaluated with a set of real BitTorrent traces.

We highlight the main contributions of this paper as follows:

1. We expand the problem space of content bundling in BitTorrent; from the currently subjective and manual bundling towards performance-driven and automatic one. We explore the potential benefits of the latter.
2. To substantiate the performance-driven and automatic bundling scheme, we propose and develop a tracker-based bundling system by modifying the Azureus software [14]. In our proposed system, the tracker can find the optimal bundle set (of torrents) that maximizes the bundling objective function with the polynomial-time complexity by modeling the optimization problem as a graph and applying a *maximum weighted perfect matching algorithm* [15] (in Section 4).
3. We suggest nine bundling strategies based on (i) the similarity among the attributes of the torrents such as titles or tags and (ii) the time-varying swarm dynamics such as popularity or availability (in Section 3), and evaluate their performance in terms of the availability and download time not only by simulation but also on real testbed (in Section 5).
4. We find that all the proposed bundling strategies outperform (by up to 50%) the no-bundling case particularly in terms of download time.
5. Among the nine suggested bundling strategies, the one that bundles more popular/available torrents with less popular/available ones outperforms others in most cases. The well-known strategy to bundle unpopular torrents (e.g. [4,7]) exhibits relatively poor performance.
6. We further reveal that title- and tag-similarity based bundling strategies provide performance improvement (in terms of download time) comparable to those of the popularity/availability based bundling strategies. This suggests that the title-similarity based bundling strategy can be a good candidate to be used in practice since it can be easily implemented in the tracker without additional effort because the title information is typically available (or easily obtainable) at the moment of content publishing.

We organize this paper as follows. After reviewing related work in Section 2, we suggest several strategies to bundle multiple torrents in Section 3. Section 4 presents a tracker-based bundling system to develop and test the proposed bundling strategies. We evaluate the bundling strategies in Section 5. After discussing other bundling issues in Section 6, we conclude the paper in Section 7.

2. Related work

2.1. Multi-torrent systems

Most studies on BitTorrent have focused on sharing a single file until Guo et al. found that 85% of users concurrently access multiple torrents [16]. Guo et al. further demonstrated the opportunity of inter-torrent collaboration; this collaboration can be effective by giving incentives to seeds if they stay longer. Yang et al. [17] also proposed an inter-torrent tit-for-tat scheme that calculates the aggregated downloading rate across all the participating torrents when unchoking, to give additional credits to users who remain as seeds in a subset of torrents, when downloading multiple torrents. Other studies have investigated more sophisticated incentive mechanisms like a credit-based scheme [18], a token-based scheme [19], propagating peer reputations [20], history-based rules [21], and a cycle-based scheme [22]. While these studies mainly have focused on incentive mechanisms leveraging the users' participation in multiple torrents, we focus on bundling, which allows peers to download multiple files from a single swarm. These studies are complementary to our work, as bundling systems may adopt above incentive mechanisms to incentivize peer collaboration.

2.2. Bundling in BitTorrent

Recently, bundling in BitTorrent has gained increasing attention as it has been reported to mitigate the availability problem of unpopular files [4,7] as well as reduce download times [4,7–10,12]. Han et al. [13] found that over 72% of BitTorrent torrents contain multiple files, which indicates that bundling is widely used. Menasche et al. [4,7] showed that bundling can mitigate the availability problem by combining multiple unpopular files into a single swarm. In particular, [7] studied the strategic interaction between a publisher who is always available (controlling prices and bundling strategies) and peers (deciding which content to download) in the context of an enterprise swarming system. While authors of [4,7] considered only a single bundling strategy to combine unpopular files, we suggest and evaluate nine bundling strategies that consider not only the user access pattern of a torrent such as its popularity or availability but also the attributes of a torrent such as its title or tags.

Tian et al. studied how to download multiple files in a bundled torrent either in a concurrent or sequential way with the assumption that files in a bundled torrent are highly interest-correlated [9]. Lev-tov et al. [8] proposed a dynamic file selection strategy (among files contained in a bundled torrent) to reduce download times, where they assumed that each peer is interested only in a small subset of the bundled files. Carlsson et al. [10] proposed a dynamic bundling scheme in which peers are assigned to download complementary files (or their chunks) at the time they decide to download a particular file to reduce the download time. Zhang et al. [12] implemented a bundling system that can support the dynamic bundling scheme [11] in practice. While these studies were looking at downloading strategies for a bundled torrent, we examine several bundling strategies and compare/evaluate them using a set of real BitTorrent traces.

2.3. Bundling in Economics

Product bundling is a common marketing strategy in economics. Bundling strategies have been proposed to increase sales, to extend monopoly power, and to smooth demands across multiple goods in the economics literature [23–26]. One strategy is to bundle unpopular products because unpopular products can be important to get profits in some businesses [27,28]. For example, Anderson [27] reported that 20% of the revenue of Rhapsody came from songs outside the top charts. Another strategy is to bundle correlated products, which is proven profitable [29]. Bakos et al. [29] reached the same conclusion in the case of information goods that have almost zero cost to be replicated. For information goods (which are digital), distribution tools such as BitTorrent already

support bundling. Mostly inspired from the bundling strategies in economics, we consider several bundling strategies in the following section.

3. (Inter-torrent) Bundling Strategies

In this section, we present several bundling strategies — what criteria can be considered in combining two torrents into a single torrent.[2] In the literature on content bundling, the well-known bundling strategy is to combine unpopular files to enhance BitTorrent performance in terms of the availability and download time of the files [4,7]. We consider various bundling strategies to combine two torrents into a single one.

Bundling strategies can be categorized into two approaches based on what criteria are used to bundle torrents: *attribute based* (i.e., using the information on or in the contents themselves – internalities) and *access-history based* (i.e., using the information external to the contents themselves – externalities). The attribute-based approach considers the attributes of a torrent such as its name, title or tags; based on the similarity of their attributes, torrents can be bundled. Because the attribute information in this approach is typically available before publishing, bundling can be done at the moment of content publication.

The access-history based approach exploits the user access pattern of a torrent such as the number of users that have accessed the torrent at a given time. For example, unpopular torrents can be bundled to enhance the BitTorrent performance in terms of the availability and download time of the torrents [4,7]; torrents can be bundled considering their correlations, found analyzing user access behaviors. Since the user access information is time-varying, access-based bundling can be done periodically (say, once in a day).

3.1. Attribute based bundling

We propose two attribute based bundling strategies in which torrents are bundled based on the similarity of their (i) title information or (ii) tag information.

Title-similarity based: Among multiple criteria for estimating the similarity of torrents (e.g., title, size, and category), title seems intuitively easiest to use. If two torrents have similar titles, then a user interested in one may also be attracted to the other (e.g., Shrek series 1, 2, 3.). To measure the similarity between the titles of torrents, we adopt a popular text classification algorithm: Levenshtein distance [30,31]. Levenshtein distance between two titles is defined as the minimum number of edits needed to transform one title into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character. Then, the title similarity of two torrents/swarms S_i and S_j is given by $title-similarity(S_i,S_j) = Levenshtein(title(S_i), title(S_j))$.

Tag-similarity based: Though the title-similarity based strategy is intuitive and easy to apply, it has limitations in some cases because the titles are not always appropriate indicators to estimate the similarity or find a relationship between contents. For example, the two songs "Billie Jean" and "Beat It" by Michael Jackson are included in the same album, but the titles are completely different. To address such cases, we propose to use "tags" when bundling, inspired from tagging systems such as those of web blogs. For example, a movie torrent can be tagged with its attribute information such as leading actors, director, distributor, genre, etc. As the current BitTorrent system does not support a tagging mechanism, we develop an automatic tagging system by which torrents are tagged with such information by leveraging existing search engines or databases, e.g., `Internet Movie Database (IMDB)` and `Amazon.com`. Our developed tagging system for a movie torrent attach 10 tags (e.g., director, distributing agency, main actors, and so on) retrieved from the `IMDB`. Once the tags of each torrent are populated, we calculate the similarity of two torrents based on the number of matched tags between two torrents/swarms S_i and S_j, denoted by $tag-similarity(S_i,S_j)$.

3.2. Access-history based bundling

We suggest four bundling strategies in the access-history based bundling category: (1) *popularity based*, (2) *availability based*, (3) *life-cycle based*, and (4) *access-correlation based*.

Popularity based: If a torrent is unpopular, users who wish to download the torrent may suffer from slow download or even unavailability. To mitigate this problem, we present two popularity based bundling strategies. Popularity of a torrent is defined as the number of currently participating users in its swarm at a given time. We consider two options in popularity based bundling: (i) to bundle two unpopular torrents and (ii) to bundle popular and unpopular torrents together. The first option to bundle unpopular content files is proposed in [4,7]. For swarms S_i and S_j, $popularity_{UU}(S_i,S_j,t) = \frac{1}{P_{S_i}(t)+P_{S_j}(t)}$ and $popularity_{PU}(S_i,S_j,t) = |P_{S_i}(t) - P_{S_j}(t)|$ are the selection criteria for the above two options (i) and (ii), respectively. Here $P_{S_i}(t)$ is the number of peers that have participated in swarm S_i until time t.

Availability based: To improve the availability of unpopular torrents by bundling, we consider two options: (i) to bundle two least available torrents and (ii) to bundle the most and least available torrents. Here, the availability of a torrent is the sum of the number of seeds at a given time, which is the widely used definition adopted in previous studies (e.g., [4]). The bundling criteria of the two options are $availability_{LL}(S_i,S_j,t) = \frac{1}{A_{S_i}(t)+A_{S_j}(t)}$ and $availability_{ML}(S_i,S_j,t) = |A_{S_i}(t) - A_{S_j}(t)|$, respectively. Here $A_{S_i}(t)$ is the availability of swarm S_i at time t.

Life-cycle based: A torrent usually has two phases over time: rising phase and falling phase [16]. Rising phase refers to the period during which the number of peers of the swarm increases (despite some fluctuations). After the rising phase is over, the number of peers decreases in the falling phase. Thus, to enhance the system throughput and/or to extend the *"lifetime"* of a torrent by bundling, we propose a bundling strategy based on the life-cycle of torrents. We estimate the life-cycle of a swarm by calculating the change of the average number of peers over time duration. That is, the life-cycle of swarm S_i from time t_0 to t is given by $Slope_{S_i}(t,t_0) = \frac{N_{S_i}(t)-N_{S_i}(t_0)}{t-t_0}$. Here $N_{S_i}(t)$ is the number of peers of swarm S_i at time t and t_0 is the time of previous bundling. If the life-cycle of a swarm is positive/negative, the swarm is in the rising/falling phase, respectively. The life-cycle based bundling has two options as well: (i) to bundle two torrents in the falling phase and (ii) to bundle torrents in the rising and falling phase. The bundling criteria of the two options are given by $life-cycle_{FF}(S_i,S_j,t,t_0) = \frac{1}{Slope_{S_i}(t,t_0)+Slope_{S_j}(t,t_0)}$ and $life-cycle_{RF}(S_i,S_j,t,t_0) = |Slope_{S_i}(t) - Slope_{S_j}(t)|$, respectively.

Access-correlation based: A more computationally-expensive bundling strategy is to analyze the correlation between two torrents in terms of user access patterns. Even if two torrents are not close in terms of title- or tag-similarity, the two torrents may be correlated if a noticeable number of users have downloaded both of them. Thus, we propose the access-correlation based bundling that considers the number of common users who have accessed both torrents by keeping track of the user access history for every torrent. That is, $access-correlation(S_i,S_j,t)$ calculates

[2] More than two torrents can be bundled by iteratively applying the bundle strategy. In this paper, we only combine two torrents for the sake of simplicity.

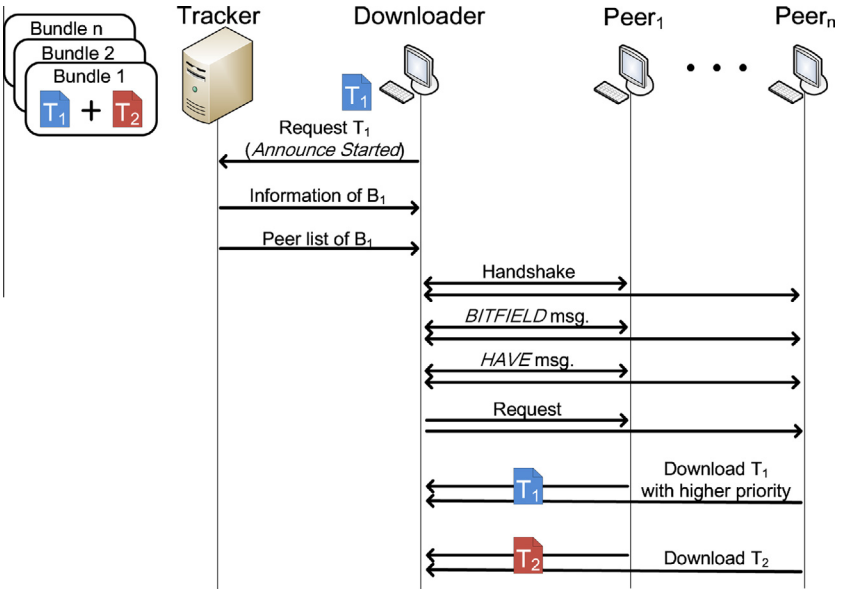


Fig. 1. The protocol description of the proposed bundling system. Bundle 1 (B_1) combines two torrents T_1 and T_2.

the number of users who have accessed both of swarms i and j until time t.

4. Tracker-based bundling system

We now detail the proposed tracker-based bundling system, in which the bundling strategies will be evaluated.

4.1. System design overview

We have three design goals for the bundling system. First, there should be little change in the BitTorrent client software, to allow for incremental deployment. Second, the system should allow users to assist in delivery of files other than those they request to improve system performance. That is, suppose a user requests torrent A, which is bundled with torrent B. The user will download not only A, but also B to assist other users or improve the system performance. Last, the service quality (i.e., the download time of the originally requested torrent) should not be degraded in our system.

We implement the bundling strategies in a tracker[3] because it already keeps track of each peer (e.g., the peer IP address and port number) in the given swarm. Hence, the tracker can easily calculate the popularity, availability, and life-cycle for the popularity based, availability based, and life-cycle based bundling strategy, respectively. In addition, the tracker can be easily extended to monitor the access-correlation between swarms since it keeps track of the peers in individual swarms. For the attribute based approach (title- and tag-similarity based bundling strategies), the tracker can easily obtain the title and tags of a torrent. Especially, tag information of each torrent can be obtained by our tagging system, which will be discussed in Section 6.

Based on the bundling objective, the tracker combines two torrents into a single one using the corresponding bundling metrics which are maintained by the tracker. Comparing all the possible bundling cases in a tracker to find the optimal set that maximizes the overall system objective is of too high complexity. To solve this problem, we apply a *maximum weighted perfect matching algorithm* [15], described below.

Fig. 1 illustrates how a user communicates with a tracker and other peers in the proposed bundling system. A downloader (with the usual BitTorrent client software) in the system accesses the tracker via conventional BitTorrent operations. Then the tracker gives information about the bundled torrent (B_1) such as content hash and a peer list of the bundled swarm to the downloader. Note that a bundled torrent consists of (i) file(s) of an original torrent (T_1) that a user requests, and (ii) file(s) of another torrent (T_2) which the user may not be interested in. Since all the files in the bundled torrent are downloaded in the proposed system, the downloader will download unwanted files as well. If priorities of the requested and unrequested files are the same, the download of the requested file(s) might be delayed. To address this issue, a higher priority is given to the requested torrent; assigning priorities to files is supported in most of BitTorrent client software. In this way, the overall swarm performance can be significantly improved without degrading the download time of the requested torrent.

We will show later that users in a bundled torrent help to improve the availability by participating in the dissemination of their unrequested file(s) (even with the lower priority). This kind of assistance or cooperation among users to improve the system performance is similar to the prior work such as [4,7–10,12,17]. In addition, participating in the cooperation will also reduce the download time of the requested file(s), which will be shown in Section 5.

4.2. Bundle formulation in a tracker

There can be many torrents/swarms managed by a tracker and hence there are numerous bundling choices among original torrents. Suppose the number of torrents/swarms of a given tracker is n and bundling of two torrents is considered. Then the number of possible bundling combinations is $\binom{n}{2} \times \binom{n-2}{2} \times \binom{n-4}{2} \cdots$. Ultimately we need to find out the set of $n/2$ bundled torrents (from n torrents) that maximizes the objective of each bundling strategy. Let us take an example of four torrents/swarms in a tracker: S_1, S_2, S_3 and S_4. There will be total three choices of bundling 4 torrents into two bundled ones: $\{S_1, S_2\}$ and $\{S_3, S_4\}$ denoted by $set1$, $\{S_1, S_4\}$ and $\{S_2, S_3\}$ denoted by $set2$, and $\{S_1, S_3\}$ and $\{S_2, S_4\}$ denoted

[3] Note that this paper considers bundling in only a single tracker. Bundling across multiple trackers are left for future work.

by *set*3. If the objective function of *set*2 is larger than that of *set*1 or *set*3, *set*2 is the optimal set that we should find.

To implement this on a tracker, we first define a matrix $Objective(S_i, S_j)$ based on the bundling metrics in Section 3, which calculates the expected objective function when two swarms S_i and S_j are bundled. We then find the optimal set (of bundled torrents) that maximizes the total sum of the objective function values (i.e., $\arg\max_{i,j}\sum_{\forall_{i,j}}^{n} Objective(S_i, S_j)$). However, finding the optimal set in an exhaustive fashion requires too much computational overhead since we need to compare all possible cases.

To solve this problem efficiently, we leverage the *maximum weighted perfect matching algorithm* [15] by converting the matrix $Objective(S_i, S_j)$ into a graph G. Let $G = (V, E, c)$ be an undirected weighted graph where c is the set of weights that are associated with edges. Each vertex represents a swarm, and each edge that connects two swarms (say S_i, S_j) means the possible bundling of the two swarms. The weight of the edge that connects S_i and S_j represents $Objective(S_i, S_j)$.

In graph theory, a *matching* is a subset of edges $E' \subseteq E$ such that no two edges in E' share a common vertex. That is, "polygamy" is not allowed [32,15]. Matching E' is *perfect* if all vertices in V are covered [15,33]. Hence, the maximum weighted perfect matching algorithm computes a perfect matching E' that has the maximum weight $c(E')$ [15,33], where $c(E')$ is the sum of weights of the edges in E'.

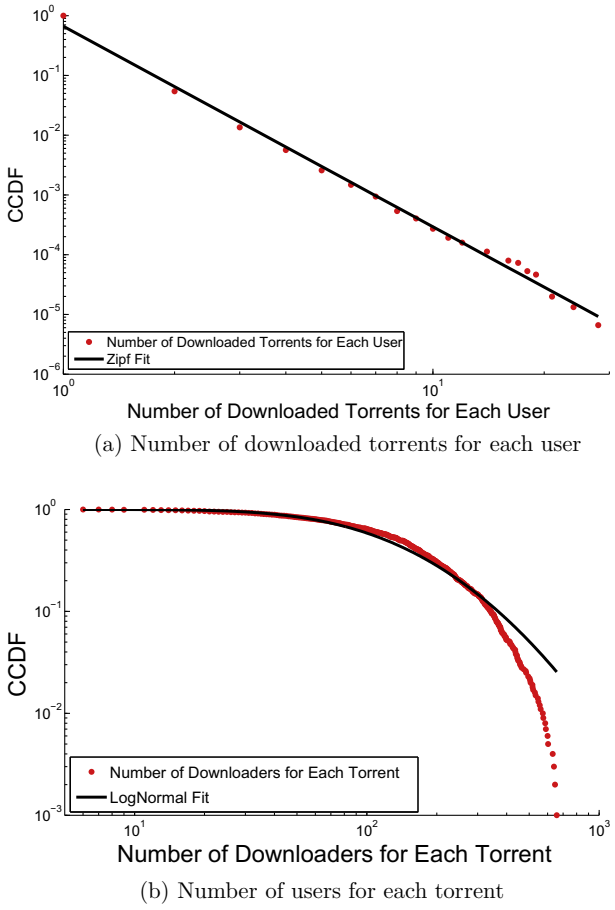
By applying the maximum weighted perfect matching algorithm, we can find the optimal set of bundled torrents. In the literature, it is known that there are algorithms to find the maximum weighted perfect matching for a general graph in polynomial time $O(n^3)$. For sparse graphs, there are faster algorithms which run within $O(nm \log n)$ where n is the number of vertices and m is the number of edges [15]. See [32,15,33] for details.

5. Evaluation

In this section, we evaluate the proposed bundling strategies with a set of real BitTorrent traces. Our experiments consist of two parts: (1) large-scale simulation and (2) experiments on a miniaturized testbed with the real BitTorrent software. Note that the large-scale simulation (which does not involve actual file transmissions) investigates various aspects of bundling effects: (i) the availability of each bundling strategy, (ii) the ratio of unrequested files of each bundling strategy, (iii) the availability and number of unrequested files when the bundle size increases, and (iv) the recommendation accept probability, based on the large-scale real traces which reflect the access patterns of peers and torrent information. On the contrary, the experiments (which involve actual file transmissions) are based on a real BitTorrent system which consists of one PC running on the Azureus [14] tracker software and 31 PCs running on the Azureus client softwares (admittedly research-grade).

5.1. Data collection

We fetched the newly published 1000 .torrent files, which were published from May 1 to May 11, 2010, from The Pirate Bay (TPB) [34], one of the most popular torrent hosting sites. To keep track of each swarm of the collected torrents, we developed a BitTorrent swarm monitoring client by modifying the Azureus software. By analyzing the ".torrent" information, the swarm monitoring clients contact tracker(s) through the tracker protocol [35] to retrieve the lists of peers. Our log data consists of 1,000 torrents, which are shared by 2,092,964 users (i.e., anonymized IP addresses). To summarize, for each torrent, we retrieve its data consisting of (i) its ".torrent" information, (ii) its content category given by TPB, and (iii) the swarm log including each peer's IP address.



(a) Number of downloaded torrents for each user

(b) Number of users for each torrent

Fig. 2. Distributions of number of downloaded torrents for each user and number of users for each torrent are plotted, respectively.

To characterize our collected data, we first plot the complementary cumulative distribution function (CCDF) of the number of downloaded torrents for each user in Fig. 2(a). The distribution is fitted to a Zipf distribution of the form $\beta x^{-\alpha}$ with parameters α = 3.324 and β = 0.632. Fig. 2(b) shows the CCDF of the number of users for each torrent, which we fit to a Lognormal distribution. The probability distribution function of the lognormal distribution is given by:

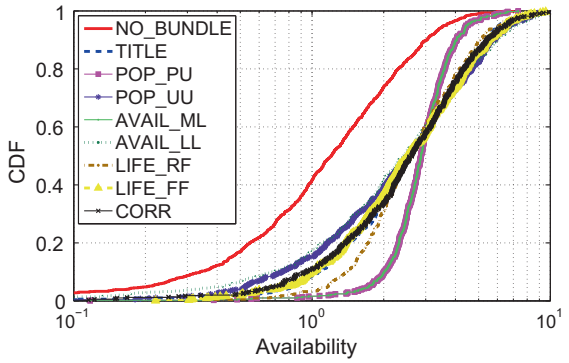
$$f(x) = \frac{1}{\sigma x\sqrt{2\pi}} e^{\frac{-(\log(x)-\mu)^2}{2\sigma^2}}$$

where parameters μ = 4.7950 and σ = 0.8672.

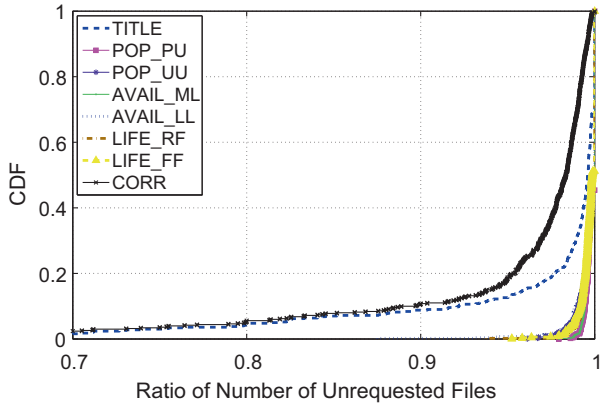
5.2. Large-scale simulation

We first conduct an extensive simulation with the log data, which consists of 1,000 torrents with 8,697,011 content requests from 2,092,964 users. Fig. 3 shows the availability and number of unrequested files when bundling strategies are applied. Note that TITLE, TAG, POP-PU, POP-UU, AVAIL-ML, AVAIL-LL, LIFE-RF, LIFE-FF, and CORR refer to *title-similarity*, *tag-similarity*, $popularity_{PU}$, $popularity_{UU}$, $availability_{ML}$, $availability_{LL}$, $life-cycle_{RF}$, $life-cycle_{FF}$, and $access-correlation$ bundling strategies, respectively. We exclude TAG in this simulation since we only have developed an automatic tagging system for Movie, TV, and E-book categories, which will be detailed later.

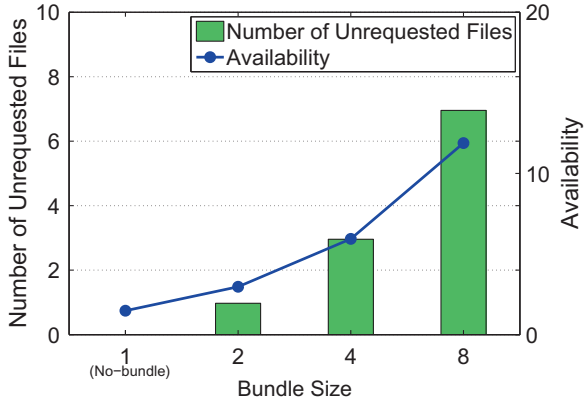
As shown in Fig. 3(a), all the bundling strategies show the better availability than the no-bundling case. Especially, POP-PU and



(a) Availability of each bundling strategy



(b) Ratio of un requested files of each bundling strategy



(c) Availability and number of unrequested files of title-similarity based bundling strategy (TITLE)

Fig. 3. Availability and ratio of the number of unrequested files to the number of total files in a bundled torrent are plotted. There is a trade-off between the availability and additionally generated traffic when the bundle size increases.

AVAIL-ML perform well for the unavailable torrents compared to the other bundling strategies. This is because POP-PU and AVAIL-ML try to raise the availability of the unavailable torrents by merging them with more popular/available ones. We believe POP-PU/AVAIL-ML can more effectively mitigate the availability problem of the swarming system than the other strategies because solving the availability problem of unavailable torrents will achieve a substantial performance improvement.

We next calculate the ratio of the number of unrequested files to the number of total files in a bundled torrent of each bundling strategy in Fig. 3(b). Recall that a user downloads unrequested file(s) in a bundled torrent in our proposed system, which implies additional traffic in the swarm. As shown in Fig. 3(b), the users in the CORR and TITLE bundling strategies download less unrequested files than those in other strategies since CORR and TITLE bundle similar or related files that both will be possibly downloaded by a given user.

We then increase the bundle size (i.e., the number of files to be bundled into a torrent) in Fig. 3(c). Note that bundling more than two torrents can be easily extended by iterative bundling. As shown in Fig. 3(c), the availability of a representative strategy, TITLE, increases as the bundle size increases, respectively. However, there is a trade-off between the availability and additionally generated traffic as the bundle size increases. That is, as the bundle size increases, users need to download more unrequested files, which will yield additional traffic. In addition, some studies like [4] show that the performance gain (i.e., download time) does not increase linearly when the bundle size increases. Therefore, we believe that the bundle size should be conservative, e.g., 2 or 4, to balance the performance gain and the additional traffic.

We next investigate another aspect of bundling: recommendation accept probability. We calculate the probability whether a user will accept the recommendation for the bundled file in Fig. 4. That is, the user downloads both of the files in the bundled torrent in the trace. Note that the recommendation accept probability is defined as $\frac{A(i)}{R(i)}$, where $R(i)$ is the number of recommended files and $A(i)$ is the number of accepted files in bundle i, respectively. As shown in Fig. 4, CORR exhibits the higher recommendation accept probability than other strategies since CORR considers the number of common users who have accessed both torrents. Interestingly, TITLE shows comparable recommendation accept probability to CORR; torrents which have similar titles are often downloaded by users.

5.3. Experiments on a real BitTorrent system

In this subsection, we conduct trace-driven experiments using a real BitTorrent system (i.e., a tracker and BitTorrent client software). To this end, we make a testbed based on the developed bundling system by modifying the Azureus [14] software. That is, the testbed consists of one PC running on the Azureus tracker software and 31 PCs running on the Azureus client softwares (admittedly research-grade). For each experiment, the number of unchoke slots, number of opportunistic unchoke slots, and maximum number of connections are set to 4, 1, and 50 respectively. Note that actual file transmissions are included in the experiments. The reason of not exploiting testbeds such as PlanetLab is that exchanging large-size files is difficult in those testbeds due to their traffic limitation (e.g., 4 GB in a day).

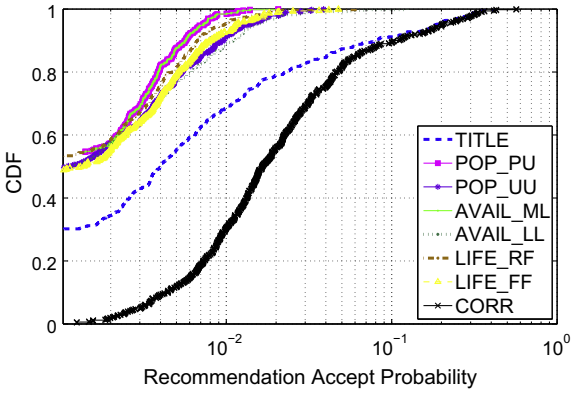


Fig. 4. Recommendation accept probability of each bundling strategy is plotted.

To emulate a real BitTorrent system in a realistic setting, we use the information of torrents and swarm dynamics from our collected log data. That is, the following characteristics of the collected data are mimicked in our emulation: (i) how many peers access each torrent, (ii) which torrents are accessed by each peer, and (iii) the torrent metadata including its title and tags. Note that tags are collected from `IMDB.com` for TV torrents and `Amazon.com` for E-book torrents through our automatic tagging system. Due to the enormous size of the real BitTorrent system in our collected data, we miniaturize each experiment with 20 torrents (before bundling). Each bundling strategy combines every pair of two torrents into a single one; thus, there will be 10 torrents (or swarms) after bundling. To select 20 torrents that can retain the characteristics of the original collected data (described in Section 5.1), we first sort all the collected torrents in descending order based on the popularity (i.e., the number of peers). Then each of the 20 torrents is randomly selected from every 50 torrents in the sorted 1,000 ones. For the purpose of fair comparison across categories, we use a fixed-size file (i.e., a 100 MB file) since the download time is proportional to the torrent size if the other conditions are equal [36].

In our experiments, the tracker first bundles torrents based on the metrics of the chosen bundling strategy. After bundling, our (developed) scheduler decides which torrents will be accessed by each peer (i.e., a BitTorrent software installed machine which is connected to the Ethernet); there are total 30 peers. Our scheduler also reflects the popularity of each torrent and the access pattern of each peer from the log data, which depends on the torrent characteristics (that can be inferred from the title and tag information). Peers arrive at a particular torrent as a Poisson process with the mean rate λ. In addition to 30 peers, there is a single source (i.e., initial seed) for each of the 20 torrents, which is alive throughout each run. After finishing the download the file(s) of interest, the peer remains in the system with the departure rate μ. Note that a peer downloads the requested file with a higher priority; thus, the other (unrequested) file will be downloaded with a lower priority.

We conduct experiments on two content categories: TV and E-book. The reason of choosing the two categories is that they exhibit two extreme access-correlation values. The access-correlation among n torrents, denoted by α, is expressed by:

$$\alpha = \frac{\sum_{\forall p \in Peers} \sum_{\forall (T_i, T_j)} I_p(T_i, T_j)}{\binom{n}{2}}$$

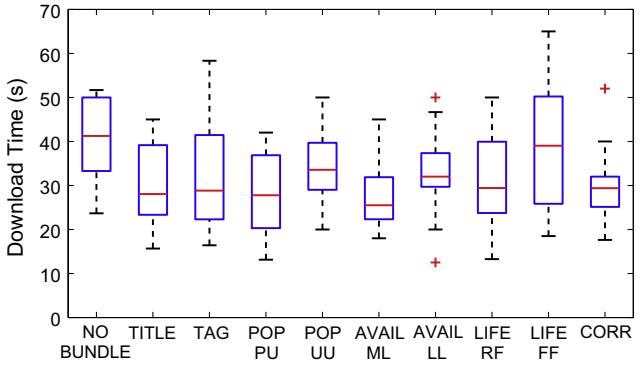
where n is the number of torrents, T_i and T_j are torrents i and j, and $I_p(T_i, T_j)$ is an indicator function whether peer p accesses both T_i and T_j or not. The α values of TV and E-book torrents are 0.011 (the smallest) and 0.268 (the largest), respectively.

5.3.1. Evaluation of the bundling strategies

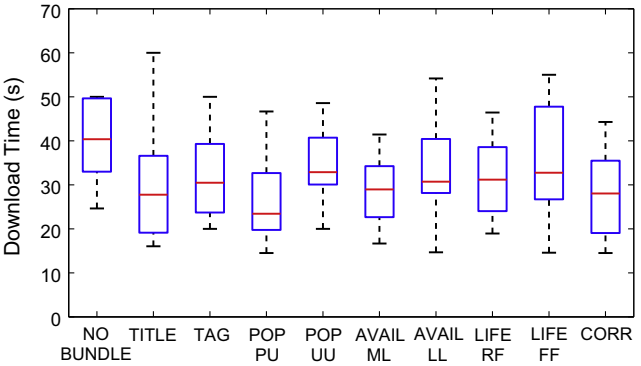
We first compare the nine bundling strategies and no-bundling when λ = 1/(3 min) and μ = 1/(5 min). Figs. 5(a) and 5(b) show the download times of the TV (α = 0.011) and E-book (α = 0.268) torrents, respectively. Note that the blue boxplots and red bars represent the distribution quartiles. Also, the black bars at the end of the dashed lines mark the 3.5th and 96.5th percentiles, respectively.

As shown in Fig. 5, all the bundling strategies outperform no-bundling. POP-PU performs substantially better than POP-UU proposed in [4,7]. Similarly, AVAIL-ML and LIFE-RF achieve notable gains over their counterparts AVAIL-LL and LIFE-FF, respectively. As POP-PU/AVAIL-ML allow a more popular/available torrent to be bundled with a less popular/available one, the users of the less popular/available torrent achieve a significant performance gain.

The download time of CORR is comparable to those of POP-PU and AVAIL-ML. This indicates that the user access correlation affects the download performance in practice. Note that CORR in the E-book category reveals a slight performance gain over CORR in the TV category since α of the E-book torrents is higher than that of the TV ones. The higher α implies the higher probability that a user requests both of the bundled files. Also, the download times of TITLE and TAG bundling strategies are close to that of CORR since bundling similar files in terms of titles or tags may reflect the user access correlation.



(a) Download time of TV torrents



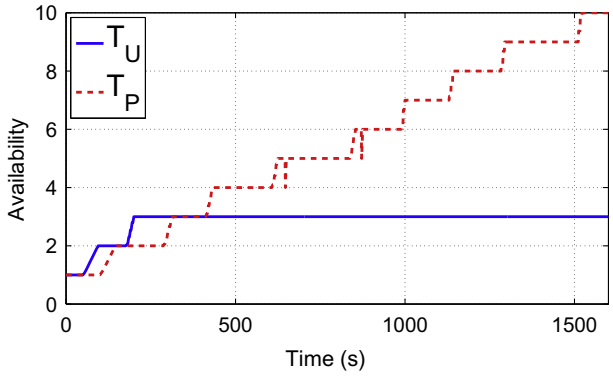
(b) Download time of E-book torrents

Fig. 5. Download time of each bundling strategy is plotted in the TV and E-book torrents/swarms, respectively.

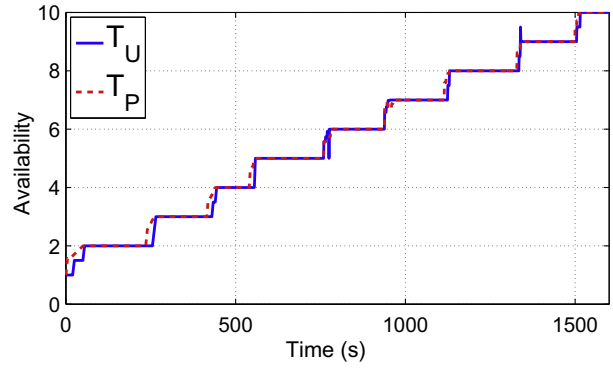
Fig. 6 illustrates the POP-PU bundling strategy and no-bundling when peers arrive with the mean arrival rate 1/(3 min) and remain (without departures) throughout each run. In both POP-PU and no-bundling cases, there are two torrents: an unpopular torrent T_U and a popular torrent T_P, which are bundled in POP-PU. The popularity values of T_U and T_P are 2 and 9 in this example, respectively. Recall that the popularity is the number of peers that access a given torrent; thus, the availability values of T_U and T_P finally reach 3 and 10 including their initial seeds, respectively. Figs. 6(a) and 6(b) show how the availability values of T_U and T_P change over time in no-bundling and POP-PU cases, respectively. As shown in Fig. 6(b), the availability of T_U in POP-PU is significantly increased because T_U is combined into the same swarm with the popular torrent T_P. The average download time of T_U in POP-PU is decreased by approximately 43% as shown in Fig. 6(c). Note that the availability of T_P in POP-PU also slightly increases due to the bundling effect, which results in 24% decrease of the download time of T_P. This implies that a more popular/available torrent can also get the benefit (i.e., higher download performance) even if it is bundled with a less popular/available torrent. This property is crucial for wider adoption of bundling, since it gives every user an incentive to participate.

5.3.2. Effect of seeding time

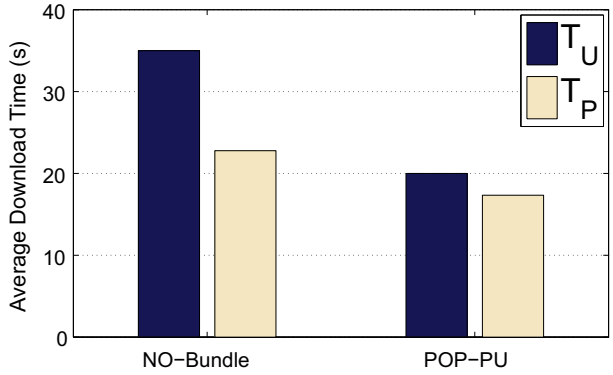
We next analyze the impact of seeding time, which refers to how long a peer remains in the system after finishing the



(a) Availability of two torrents (T_U and T_P) in no-bundling case



(b) Availability of two torrents (T_U and T_P) in POP-PU case

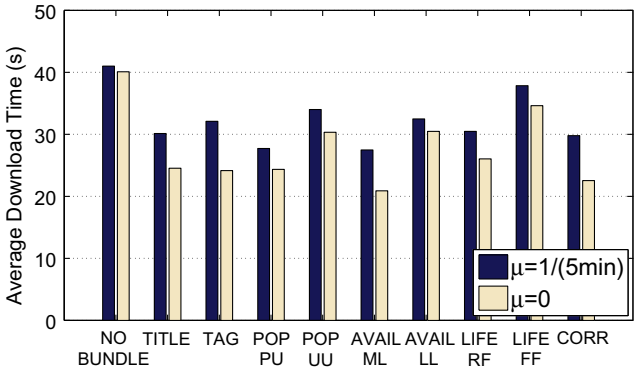


(c) Average download times of two torrents (T_U and T_P) in no-bundling and POP-PU cases

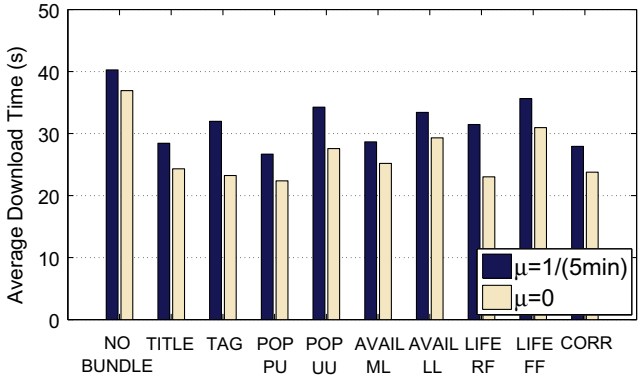
Fig. 6. Availability and download times of torrents T_U and T_P are plotted, where T_U and T_P are unpopular and popular torrents, respectively. POP-PU substantially decreases not only the download time of T_U, but also that of T_P, compared to no-bundling.

download. That is, we vary the departure rate μ from 1/(5 min) to zero (i.e., peers stay forever).

Fig. 7 compares the average download times when μ varies from 1/(5 min) to zero. The average download times decrease notably as μ becomes zero, since more seeds stay in the system over time. Note that the performance gains of the bundling strategies due to longer seeding time are significant compared to that of no-bundling. The download time of AVAIL-ML in case of μ being zero is almost half of that of no-bundling in the TV torrents, which highlights bundling along with long seeding time can enhance the system performance substantially.



(a) Average download time of TV torrents



(b) Average download time of E-book torrents

Fig. 7. Impact of seeding time on the system performance is shown when the departure rate changes from 1/(5 min) to zero.

In summary, among the suggested bundling strategies, the one that bundles more popular/available torrents with less popular/available ones outperforms others in most cases; for the performance-driven purpose, these bundling strategies can be a good option. Interestingly, the title-similarity based bundling can provide performance improvement comparable to those of the popularity/availability based bundling strategies and higher recommendation accept probability. This signifies that the title-similarity based bundling can be a good candidate to be used in practice since it can be easily implemented in the tracker without additional effort and applied because this can be done offline when the files are published because the title information is typically available at the time of publishing. We will investigate the title-similarity based bundling in Section 6.

6. Discussions

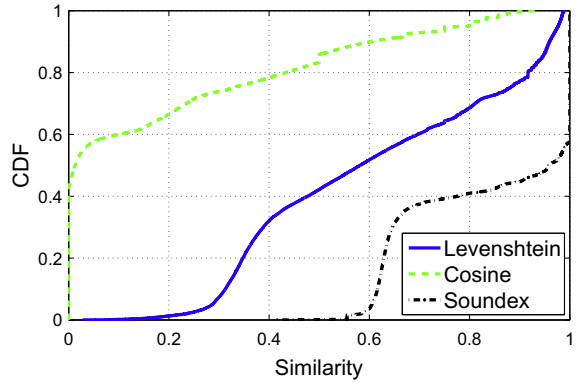
Title-similarity: When we estimate the similarity of two torrents by comparing their titles, we use the Levenshtein algorithm. There are many string matching algorithms in the literature [30,31]. To find out the effect of the string matching algorithms on the title-similarity, we evaluate three representative algorithms: (i) *Levenshtein*, (ii) *Cosine Similarity* in which two titles are transformed into two vectors (where each word of a title is projected into each axis) and the cosine function of the angle between the two vectors quantifies the similarity, (iii) *Soundex* which is a phonetic algorithm for indexing names by sound, as pronounced in English; vowels and numbers are not counted and consonants with the similar sound are counted equal. For example of two strings "Robert" and "Rupert," the title-similarity calculated by

Levenshtein, Cosine Similarity, and Soundex is 1/3, 0, and 1, respectively.

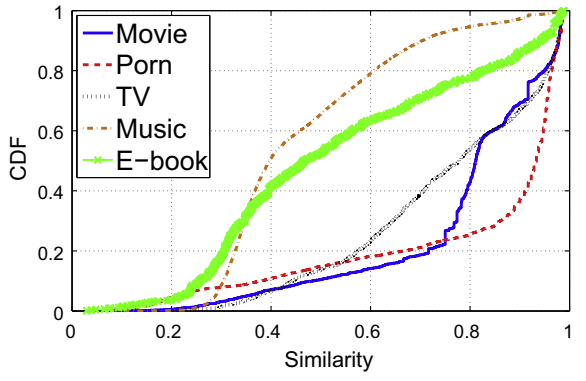
Sometimes, a user bundles multiple files into a single torrent voluntarily (not by the proposed bundling strategy). There are two cases of user bundling: (i) a main file and other supplementary files, and (ii) multiple main files and possibly other supplementary files. Here a main file is the primary file of interest to potential downloaders. If a user bundles video files of multiple episodes of a TV drama, that is the latter case.

To investigate whether main files in a torrent bundled by a user have similar titles, we crawled all the published ".torrent" files on TPB from April 22 to May 29, 2010. Among collected 110 K torrents, we use 59 K bundled torrents of the five major content categories given by TPB: Movie, Porn, TV, Music, and E-book. We consider only the primary media files as main files of a torrent; other supplementary files such as caption or trailer in the torrent are not counted as main files. For example, a ".avi" file in a Movie torrent or a ".mp3" file in a Music torrent is the main file. Note that there may be multiple main files in a bundled torrent; if a publisher bundles two episodes of a TV drama, there are two main files in the bundled torrent.

We first compare how the three algorithms quantify the title-similarity in a user bundling torrent in our datasets as shown in Fig. 8(a). For the bundled torrents in TV, two different episodes of the same season in the same TV series exhibit high similarity (around 0.9) by Levenshtein. If two different episodes belong to different seasons but to the same TV series, the similarity result by Levenshtein is around 0.8. Since the titles of main files for the same series are typically named with minor variations (e.g., different episode indices) by users, Levenshtein can calculate the title similarity with fine granularity. However, since a user often writes the words of a title without spacing, Cosine Similarity cannot demarcate the words of the title and hence its title-similarity is calculated often zero. Thus it may not be effective with manually written titles with no spacing. Soundex also has drawbacks because it often regards two titles as the same one if the two titles have different episode numbers, or are phonetically similar (e.g., Star Wars 1 and Star Wars 2). Therefore, we adopt the Levenshtein algorithm to estimate the similarity of two files by comparing their titles.



(a) Title-similarity for the three string matching algorithms



(b) Title-similarity among main files in a torrent

Fig. 8. CDF of title-similarity for the representative string matching algorithms and among main files in a torrent (bundled by users) are plotted, respectively.

We next analyze the similarity among the titles of main files in a user bundling torrent across the five different content categories. We calculate the title-similarity using the Levenshtein algorithm. Fig. 8(b) shows that the CDF of title-similarity of the Porn torrents is significantly higher than those of the others (80% of the Porn torrents have the similarity values higher than 0.7). This is mainly due to the file naming convention of (i) pornographic pictures containing special words such as publisher names like "Met-Art", and (ii) two video files (in two discs) having the same title except the suffixes of CD1 and CD2. The title-similarity values of the TV and the Movie torrents are also high because titles of the episodes of the same series are usually similar. On the contrary, the title-similarity values of the E-book and the Music torrents are lower due to diverse titles. So content categories need to be carefully considered when we adopt the title-similarity based bundling strategy.

Tag-similarity: The tag-similarity based bundling strategy requires tags describing the characteristics of the main files. Because user-generated tags may be unreliable and often unavailable, we develop an automatic tagging system that attaches tags to a file. The automatic tagging system can rely on different legacy systems depending on the content category. For example, for Movie and TV torrents, the tagging system can send queries to the `IMDB.com`. In the music category, the tagging system may exploit a search engine specialized in music, or retrieve the metadata at the front part of an mp3 file. For the E-book contents, `Amazon.com` can be used as a book database. After obtaining the tags, we can calculate the tag-similarity based on the number of matched tags.

Incentives of users in the tracker-based bundling system: A bundled torrent in the proposed system consists of (i) files that a user requests, and (ii) files that the user may not be interested in. Since all the files in a torrent are downloaded, the user may download unrequested files as well. This raises the following question: why users would be willing to participate in the proposed system? First, the overall swarm performance can be significantly improved without degradation of the download time of the requested file. By giving a higher priority to the requested torrent file(s), the download time of the requested torrent is not degraded in the proposed bundling system. As shown in Section 5, users in a bundled torrent help to improve the overall swarm performance by participating in the dissemination of their unrequested file(s). This kind cooperation among users to improve the system performance is similar to the prior studies such as [4,7–10,12,17]. Second, participating in the cooperation will also reduce the download time of the requested file(s) as shown in Section 5, which can be a clear incentive to participate in the proposed system. Interestingly, a more popular/available torrent can also get the benefit (i.e., higher download performance) even if it is bundled with a less popular/available torrent as shown in Section 5. This property is crucial for wider adoption of bundling, since it gives every user an incentive to participate. Finally, the proposed bundling system can additionally provide the content recommendation services. In our trace-driven simulation, we showed that the recommendation accept probability of the title-based bundling strategy is higher than those of other bundling strategies (except the access-correlation based bundling which requires additional efforts to keep track of the each user's access history).

7. Conclusions

While the current bundling practice in BitTorrent is mostly done in a subjective and manual manner by individual publishers, we believe it is worthwhile to explore the potential benefits of automatic and performance-driven bundling. To this end, we came up with nine systematic bundling strategies considering the attributes of the torrents and the time-varying swarm dynamics. To substantiate the automatic and performance-driven bundling scheme, we proposed and developed a tracker-based bundling system, where all the proposed bundling strategies are implemented and evaluated with a set of real BitTorrent traces. We found that all the proposed bundling strategies outperform no-bundling case in terms of the availability and file download time. Among the nine suggested bundling strategies, the one that bundles more popular/available torrents with less popular/available ones outperforms others in most cases; for the performance-driven purpose, the above bundling strategies can be a good option. We also found that title- and tag-similarity based bundling strategies provide performance improvement (in terms of download time) comparable to those of the popularity/availability based bundling strategies. This suggests that the title-similarity based bundling can be a good candidate to be used in practice since it can be easily implemented in the tracker without additional effort because the title information is typically available at the moment of publishing. We considered nine bundling strategies in this paper, but mixing different bundling strategies will bring up challenging issues such as: (i) how users will participate in a swarm with combined bundling strategies and (ii) how to find out the bundling combination that maximizes the multiple bundling objectives with low algorithmic complexity. Another imminent topic is to study how to dynamically bundle and unbundle files to adapt to time-varying system dynamics (e.g., the popularity of torrents, user churning).

Acknowledgement

This research was supported in part by Basic Science Research Program through the "National Research Foundation of Korea (NRF)" funded by the Ministry of Science, ICT & future Planning (2013R1A2A2A01016562) and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (No. 2013-R1A1A2010474).

References

[1] Sandvine global internet phenomena report – 2h 2012. URL <http://www.sandvine.com/news/global_broadband_trends.asp>.
[2] D. Qiu, R. Srikant, Modeling and performance analysis of bittorrent-like peer-to-peer networks, in: ACM SIGCOMM, 2004.
[3] A. Legout, G. Urvoy-Keller, P. Michiardi, Rarest first and choke algorithms are enough, in: ACM IMC, 2006.
[4] D.S. Menasche, A.A. Rocha, B. Li, D. Towsley, A. Venkataramani, Content availability and bundling in swarming systems, in: ACM CoNEXT, 2009.
[5] G. Dán, N. Carlsson, Dynamic swarm management for improved bittorrent performance, in: IPTPS, 2009.
[6] G. Neglia, H. Zhang, D. Towsley, A. Venkataramani, J. Danaher, Availability in bittorrent systems, in: IEEE INFOCOM, 2007.
[7] D.S. Menasche, G. Neglia, D. Towsley, S. Zilberstein, Strategic reasoning about bundling in swarming systems, in: IEEE GameNets, 2009.
[8] N. Lev-tov, N. Carlsson, Z. Li, C. Williamson, S. Zhang, Dynamic file-selection policies for bundling in bittorrent-like systems, in: IEEE IWQOS, 2010.
[9] Y. Tian, D. Wu, K.-W. Ng, Analyzing multiple file downloading in bittorrent, in: IEEE ICCP, 2006.
[10] N. Carlsson, D.L. Eager, A. Mahanti, Using torrent inflation to efficiently serve the long tail in peer-assisted content delivery systems, in: IFIP Networking, 2010.
[11] S. Zhang, N. Carlsson, D. Eager, Z. Li, A. Mahanti, Towards a dynamic file bundling system for large-scale content distribution, in: IEEE MASCOTS, 2011.
[12] S. Zhang, N. Carlsson, D. Eager, Z. Li, A. Mahanti, Dynamic file bundling for large-scale content distribution, in: IEEE LCN, 2012.
[13] J. Han, S. Kim, T. Chung, T.T. Kwon, H. Kim, Y. Choi, Bundling practice in bittorrent: what, how, and why, in: ACM SIGMETRICS, 2012.
[14] Open sourced bittorrent client, vuze. URL <http://www.vuze.com>.
[15] Z. Galil, Efficient algorithms for finding maximum matching in graphs, ACM Comput. Surv. 18 (1) (1986) 23–38.
[16] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, Measurements, analysis, and modeling of bittorrent-like systems, in: ACM IMC, 2005.
[17] Y. Yang, A.L.H. Chow, L. Golubchik, Multi-torrent: a performance study, in: IEEE MASCOTS, 2008.
[18] M. Sirivianos, J. Han, P. Rex, C.X. Yang, Free-riding in bittorrent networks with the large view exploit, in: IPTPS, 2007.
[19] A. Ramachandran, A. das Sarma, N. Feamster, Bitstore: an incentive compatible solution for blocked downloads in bittorrent, in: NetEcon, 2007.
[20] M. Piatek, T. Isdal, A. Krishnamurthy, T. Anderson, One hop reputations for peer to peer file sharing workloads, in: USENIX NSDI, 2008.
[21] N. Carlsson, D.L. Eager, Modeling priority-based incentive policies for peer-assisted content delivery systems, in: IFIP Networking, 2008.
[22] R. Eidenbenz, T. Locher, S. Schmid, R. Wattenhofer, Boosting market liquidity of peer-to-peer systems through cyclic trading, in: IEEE P2P, 2012.
[23] W.J. Adams, J.L. Yellen, Commodity bundling and the burden of monopoly, Q. J. Econ. 90 (3) (1976) 475–498.
[24] M.A. Salinger, A graphical analysis of bundling, J. Business 68 (1) (1995) 85–98.
[25] R. Fuerderer, A. Herrmann, G. Wuebker, Optimal Bundling, Springer, 1999.
[26] R.B. Wilson, Strategic models of entry deterrence, Handbook Game Theory Econ. Appl. 1 (1) (1992) 305–329.
[27] C. Anderson, More long tail debate: mobile music no, search yes, 2008. URL<http://www.thelongtail.com/>.
[28] C. Anderson, The Long Tail: Why the Future of Business is Selling Less of More, Hyperion, 2006.
[29] Y. Bakos, E. Brynjolfsson, Bundling information goods: pricing, profits, and efficiency, Manage. Sci. 45 (12) (1999) 1613–1630.
[30] G. Navarro, A guided tour to approximate string matching, ACM Comput. Surv. 33 (1) (2001) 31–88.
[31] J. Lewis, S. Ossowski, J. Hicks, M. Errami, H.R. Garner, Text similarity: an alternative way to search medline, Bioinformatics 22 (18) (2006) 2298–2304.
[32] J. Edmonds, Paths, trees, and flowers, Can. J. Math. 17 (1965) 449–467.
[33] V. Kolmogorov, Blossom v: a new implementation of a minimum cost perfect matching algorithm, Math. Program. Comput. 1 (1) (2009) 43–67.
[34] The pirate bay. URL <http://thepiratebay.org/>.
[35] Bittorrent tracker protocol. URL <http://wiki.theory.org/BitTorrent_Tracker_Protocol>.
[36] W. Galuba, K. Aberer, Z. Despotovic, W. Kellerer, Leveraging social networks for increased bittorrent robustness, in: IEEE CCNC, 2010.