Caching-Based Multi-Swarm Collaboration for Improving Content Availability in BitTorrent

HyunYong LEE†‡, Masahiro YOSHIDA†§, Nonmembers, and Akihiro NAKAO††, Member

SUMMARY Despite its great success, BitTorrent suffers from the content unavailability problem where peers cannot complete their content downloads due to some missing chunks, which is caused by a shortage of seeders who hold the content in its entirety. The multi-swarm collaboration approach is a natural choice for improving content availability, since content unavailability cannot be overcome by one swarm easily. Most existing multi-swarm collaboration approaches, however, suffer from content-related limitations, which limit their application scopes. In this paper, we introduce a new kind of multi-swarm collaboration utilizing a swarm as a temporal storage. In a nutshell, the collaborating swarms cache some chunks of each other that are likely to be unavailable before the content unavailability happens and share the cached chunks when the content unavailability happens. Our approach enables any swarms to collaborate with each other without the content-related limitations. Simulation results show that our approach increases the number of download completions by over 50% (26%) compared to normal BitTorrent (existing bundling approach) with low overhead. In addition, our approach shows around 30% improved download completion time compared to the existing bundling approach. The results also show that our approach enables the peers participating in our approach to enjoy better performance than other peers, which can be a peer incentive.

key words: multi-swarm collaboration, content unavailability, BitTorrent

1. Introduction

BitTorrent [1] has become one of the successful applications for scalable content distribution over the Internet and it accounts for roughly 30% of all traffic on the Internet [2]. BitTorrent, however, suffers from several limitations including content unavailability where the peers cannot receive a content in its entirety from the chunks available in a swarm. The content unavailability problem, which is caused by a shortage of seeders who hold the content in its entirety, prevents the peers from completing content download. Existing work [5] and the measurement data we present in this paper show that the content unavailability problem occurs on a huge scale. For example, 40% of the swarms have no seeder available for more than 50% of its swarming period [5].

Several works exploit the multi-swarm collaboration approach to improve the content availability, since the content unavailability cannot be overcome by one swarm easily. Although existing work improves the content availability, they suffer from some content-related limitations, which limit their application scopes. For example, Guo et al. [3] and Yang et al. [4] enable a peer to share its downloaded contents with other swarms as a seeder while acting as a leecher in its original swarm. This approach requires the downloaded contents to be shared across swarms. Moreover, there should be an appealing incentive to encourage the peers to share the downloaded contents across swarms. Mensache et al. [5] tries to let peers remain longer in a swarm by enlarging swarm size through a bundling of a number of related contents. The content bundling only can be applied to a set of related contents like TV show series. The peers may also need to download content that they do not want in order to download content that they want.

In this paper, we try to improve the content availability based on the multi-swarm collaboration from a somewhat different aspect. We enable the swarms to make the multi-swarm collaboration by regarding the counterpart swarm as a temporal storage unlike most existing work trying to make the multi-swarm collaboration based on the available contents. In a nutshell, the collaborating swarms cache some chunks of each other that are likely to be unavailable before the content unavailability happens (i.e., when seeders are online) and share the cached chunks when the content unavailability happens. With this approach, we intend to provide whole chunk set through help of the collaborating swarm after the seeder leaving. This approach allows any swarms to collaborate with each other without the content-related limitations.

We conduct extensive measurements including 1.5 million swarms to study the content unavailability problem. From the measurement, we find that 72.6% of swarms suffer from the content unavailability and that small swarms are more likely to face content unavailability. We conduct simulations to examine our multi-swarm collaboration approach. Through simulations, we find that the number of download completions can be increased by over 50% with low caching overhead compared to normal BitTorrent (e.g., the highest caching overhead is 0.22 chunks per peer). Our approach also improves the number of download completions by over 26% (at most 344%) and the download completion time by around 30% compared to the existing bundling approach [5] under certain environment. We also find that our approach enables the peers participating in our approach to enjoy better download completion time than other peers, which can be a peer incentive. The simulation results show that...
the increase of caching overhead (i.e., a number of cached chunks) does not lead to linear increase of the performance while the more number of cached chunks usually results in better performance. Therefore, as future work, we plan to study better way to select the chunks to be cached to have better performance improvement and efficient chunk caching approach to cache more chunks.

The remainder of the paper is organized as follows. Section 2 introduces our measurement result and related work. Section 3 discusses the proposed multi-swarm collaboration scheme in detail. Section 4 evaluates our approach and Sect. 5 concludes this paper.

2. Content Unavailability and Multi-Swarm Collaboration

2.1 Terminologies in BitTorrent

For clear presentation, we describe BitTorrent terms used in this paper. Swarm indicates a set of peers sharing the same content. Leecher is any peer that does not have the entire content and is downloading the content. Seeder is a peer who has 100% of the content. When a leech obtains 100% of the content, that peer automatically becomes a seeder. Tracker is a server that keeps track of which peers are in the swarm. Peers report information to the tracker periodically and receive information about other peers to which they can connect. Bitfield message is exchanged between the peers to show a possession of specific part of content.

2.2 Content Unavailability

In BitTorrent, content is divided into segments, called chunks. Each swarm consists of leechers or/and seeders. The peers exchange chunks with each other using a tit-for-tat strategy (i.e., exchanging chunks preferentially with other peers whom they have successfully exchanged chunks in the past at high bandwidth) until they complete their downloads. Content is available if at least one seeder is online or remaining leechers can collectively make all chunks available [10]. If the leechers can not construct original content from available chunks, we say that the content unavailability problem happens and this is mostly caused by a shortage of seeders. Seeders may become unavailable due to several reasons [5]. Content publishing sites serving a large number of contents may take down the seeders after the initial distribution in order to reduce their costs. Even though the leechers can become the seeders when they download all chunks, they usually leave the swarm soon after completing their downloads [10].

We conduct extensive measurements including 1.5 million swarms to understand the content unavailability. Here, we show main observations. Detailed method of the measurement is described in [12]. From the measurement, we find that just 15% of swarms (i.e., 232,636 swarms) have more than 1 seeder. In this case, the swarm size ranges up to 58,980. On the other hand, 72.6% of swarms suffer from the content unavailability and 12.4% of swarms (with only one seeder) are likely to face the content unavailability. In this case, the swarm size ranges up to less than 1,000 and 99% of the swarms are with less than 30 leechers. This result shows that the content unavailability problem happens on a huge scale and leads us to study a way for improving the content availability, which may enhance the performance of most swarms.

2.3 Existing Multi-Swarm Collaboration Approaches

Until now, little research work has been performed to solve the content unavailability problem of BitTorrent while most existing work exploits the multi-swarm collaboration. Guo et al. [3] were first to propose an idea for the multi-swarm collaboration. Based on their extensive measurements and analytical models, they introduce the tracker site overlay allowing peers to share downloaded contents across swarms. They show that swarm lifetimes can be extended if a peer that acts as a leecher in one swarm also acts as a seeder in another swarm. Following this, Yang et al. [4] tries to provide an appealing incentive that makes peers to share their downloaded contents with other swarms while acting as leechers in one swarm. As the incentive, they propose a cross-swarm tit-for-tat scheme in which unchoking (i.e., uploading chunks to a peer who issues a content request) is done based on the aggregate download rate from each candidate peer across all swarms. Other several work also tries to make inter-swarm content exchange happens by proposing various incentive mechanisms such as incentive-based token scheme [7], history-based priority scheme [8], and peer reputation propagation [9].

Unlike above work trying to overcome the content unavailability based on the exchange of downloaded contents, some work tries to improve the content availability by enlarging swarm size. Menasche et al. [5] tries to improve the content availability by bundling a number of related contents to prolong online time of peers. It is shown that the bundling of an appropriate number of contents can reduce the content unavailability period in swarms with highly unavailable seeders. Dan et al. [6] shows how multiple swarms of the same content can be merged into larger swarm to improve the performance of small swarms. By adaptively merging swarms, the small swarms that are sensitive to fluctuations in the peer participation can improve performance as well as the content availability.

On the other hand, rather than depending on the available contents (i.e., downloaded contents, contents to be bundled, and contents being distributed through multiple swarms), this paper tries to improve the content availability without the content-related limitations by utilizing a swarm as temporal storage. Therefore, any swarms with appropriate amount of resources required for the collaboration (e.g., storage and upload bandwidth) can enjoy improved content availability with our approach without any content-related limitations.
3. Temporal Caching-Based Multi-Swarm Collaboration

Our goal is to improve the content availability by enabling the collaborating swarms to cache some chunks that the counterpart swarm may not be able to provide when the seeder goes offline. Thus, our main challenges include start/end time of the collaboration, selection of swarms to be matched for the collaboration, and selection of chunks to be cached. We will discuss these issues in this Section and corresponding state diagram is depicted in Fig. 1. From now on, for the sake of simplicity, we call our approach as MSC shortly.

We assume that an original content publisher uploads at least one complete copy of content before leaving the swarm. We also assume that a set of swarms to be considered for the multi-swarm collaboration is given. We define the content availability as follows. Let assume a swarm $S_i$ managing the content composed by $C_i$ chunks. We define the function of $V_m$, $V_m = [V_{m1}, V_{m2}, \ldots, V_{mC}]$ of peer $m$, where $V_{mn}$ is all the chunk $n$ or 0 otherwise. The bitfield of peer $m$ can be represented as $V_m$ in BitTorrent. Thus, the content availability, or percentage of available chunks of swarms $S_i$ is

$$A(S_i) = \frac{\sum_{m=1}^{C_i} OR(V_{mn})}{C_i},$$

where $OR(V_{mn})$ is OR-operation over the chunk $n$ across all the peers in the swarm $S_i$. We say that the content unavailability happens when $A(S_i) < 1$.

Tracker-based swarm monitoring (BitTorrent Swarming): MSC utilizes the tracker, since the tracker is at the best position to understand overall status of the swarm. For the swarm monitoring, peers report bitfield to the tracker periodically. To reduce the reporting overhead, the peers can attach the bitfield to the existing Tracker Request message that is sent by the peers to the tracker periodically. Based on the peer report, the tracker monitors the content availability of its swarms and takes appropriate actions (of Fig. 1) periodically.

Multiple trackers can build a tracker overlay for the collaboration between swarms managed by different trackers. A list of trackers to be connected can be given by content providers or the peers can report the list of trackers that they have contacted when they access to their trackers. A specific method for overlay construction and management is out of the scope of this paper but [11] may be used.

Swarm selection (BitTorrent Swarming): The tracker needs to select swarms to be matched for the collaboration based on the monitoring result. The tracker starts MSC for $S_i$ when there are the seeders and the seeders are likely to leave the swarm soon. For this, we use following conditions as decision criteria: $A(S_i) = 1$ and $A(\tilde{S}_i) > \alpha$, where $A(\tilde{S}_i)$ is the content availability of $S_i$ except the seeders and $\alpha$ is $min(1, D + \text{seeder/leecher ratio})$ ($D < 1$). $A(\tilde{S}_i)$ indicates an estimated content availability without the seeders. $A(\tilde{S}_i)$ increases when the seeders are uploading the content (since a number of available chunks at leechers increases) and $A(\tilde{S}_i)$ is close to 1 when the seeders complete the content upload (since the leechers may have almost chunks collectively). Therefore, $A(\tilde{S}_i)$ may be equal or close to 1 when the seeders are likely to leave the swarm after completing content upload. On the other hand, $\alpha$ indicates a current seeding status and changes with the number of seeders. When there are enough seeders, the decision criteria cannot be met, since $\alpha$ may equal to 1 while $A(\tilde{S}_i)$ cannot over 1. If $\alpha$ decreases below 1 due to the seeder leaving, the decision criteria may be satisfied, since $A(\tilde{S}_i)$ may still close to 1 as long as there exist some seeders regardless of number of seeders. Thus, the decision criteria allows the swarms with few seeders to be selected for MSC.

$D$ is closely related to the satisfaction of decision criteria. If $D$ is low, the decision criteria can be satisfied even with many seeders. On the other hand, the decision criteria can barely be satisfied with few seeders if $D$ is close to 1. Therefore, $D$ can be adjusted to a feature of seeders of given swarm. For example, if the seeders are highly dynamic, $D$ can be set to low so that enough chunks can be cached before the seeders go offline. We do not discuss a way to determine appropriate $D$ according to given features of seeders due to the limit of space and we leave it for future work. $D$ also affects the number of chunks to be cached, since it determines the start time for the collaboration. For example, the lower $D$, the more number of chunks to be cached. We will discuss how $D$ affects the collaboration performance and overhead later.

Swarm matching (Messaging I, Swarm Matching): Basically, two swarms are matched for the collaboration in our approach. With our approach, any two swarms can be matched, since we do not need to care about the available contents for the swarm matching unlike existing approaches. Instead, we focus on fair collaboration between two swarms.

Fig. 1 State diagram of the proposed multi-swarm collaboration.
(e.g., the collaborating swarms contribute similar resources including storage and upload bandwidth for the collaboration). For this, we define rare chunks as the chunks that are likely to be unavailable when the seeders and the leechers who have larger than $\beta$% of whole chunks leave the swarm. Here, we care about the leechers in addition to the seeders, since they may leave the swarm soon after completing the content download[10].

Based on the number of rare chunks, the swarm matching procedure is done as follows. When $A(S_j) = 1$ and $A(S_j) > \alpha$, a corresponding tracker, let say A sends MSCSearch($e_i$, $S_i$) to other trackers through the tracker overlay after identifying the number of rare chunks of $S_i$, $e_i$. Other trackers respond it by returning MSCInterested($e_j$, $S_j$, $S_i$), where $S_j$ is a swarm that has smallest $|e_i - e_j|$ among their swarms. Among the received MSCInterested() messages, the tracker A chooses one swarm that has smallest $|e_i - e_j|$. If $e_i > e_j$, ($e_i > e_j$), $S_i (S_j)$ caches some rare chunks of $S_j (S_i)$ (e.g., chunks that have low replication ratio) multiple times so as to match total number of chunk cachings. By doing this, two collaborating swarms cache same number of rare chunks of each other.

$\beta$ affects a number of rare chunks. For example, the lower $\beta$, the more number of rare chunks. Like the case of $D$, $\beta$ can be adjusted according to a feature of leechers of given swarm. Together with $D$, we leave a way to determine appropriate $\beta$ for future work. Instead, we will show how $\beta$ affects the collaboration performance and overhead later.

**Chunk caching (Chunk Caching):** After the swarm matching (e.g., $S_i$ and $S_j$ are matched), corresponding trackers (e.g., A and B, respectively) exchange a list of leechers that will cache the rare chunks. Then, the trackers let their peers who have the rare chunks send the rare chunks to the leechers of counterpart swarm. The leecher to transfer the chunk to be cached is randomly selected among the candidate leechers. Each selected peer transfers at most one chunk to be cached for each interval not to degrade its swarming performance. If there are multiple peers having the same rare chunk, the tracker randomly selects one peer, since we believe that the upload of one chunk does not much degrade the performance of the peer. For stable chunk caching, a leecher who has smallest number of chunks is selected to cache each rare chunk, since it is more likely to stay longer in a swarm than others until completing its content download. If there are multiple leechers with the smallest number of chunks, the tracker randomly selects one leecher. If the number of rare chunks is larger than a swarm size, leechers needs to cache more than one chunk. In this case, for each rare chunk to be cached, a leecher who has smallest number of chunks among leechers caching smallest number of rare chunks is selected for the chunk caching. With this approach, caching overhead can be distributed evenly over leechers.

To cope with a dynamic nature of peers, the tracker selects another leecher with the peer selection way explained above when the leecher caching the chunk is likely to leave the swarm (i.e., when it downloads larger than $\beta$% of whole chunks). Then, the tracker lets the newly selected leecher caches the chunk by downloading it from the previous leecher.

**Chunk sharing (Messaging II, Chunk Sharing):** The tracker begins to share the cached chunks when one of the collaborating swarms faces the content unavailability (i.e., $A(S_j) < 1$ or $A(S_j) < 1$). When $A(S_j) < 1$, for example, the tracker A sends MSCRequest(a list of missing chunks, leechers who will receive the cached chunks, $S_j$) to the tracker B managing the counterpart swarm $S_j$. In return, the tracker B sends MSCResponse() including the same kind of information with MSCRequest(). One swarm (e.g., $S_j$) may not face the content unavailability when the chunk sharing begins, since the chunk sharing begins when one of the collaborating swarms faces the content unavailability. In this case, $S_j$ can utilize the cached chunks for better chunk replication. In case of $S_j$, the missing chunks are chunks that have low replication ratio among the cached chunks. Here, a number of missing chunks of the collaborating swarms are same for the fair collaboration. After exchanging of request/response messages, the trackers let their leechers send the cached chunks once to the counterpart swarm so that the missing chunks are available in the counterpart swarm. Basically, the leechers caching the rare chunks receive the cached chunks first. When the leechers receive the cached chunks from the counterpart swarm, they share the received chunks with the leechers caching chunks first and then others. With this, we intend to provide better performance to the leechers participating in the chunk caching than others as an incentive. In addition, for better chunk replication, the leechers upload the received chunks to at most two leechers when they receive the cached chunks.

Sharing chunks across swarms stops when the cached chunks are not required anymore and MSC re-start from Chunk Caching (if the content unavailability problem is solved through sharing the cached chunks and thus no more sharing is required) or MSC is over and each swarm transits to BitTorrent Swarming (if the content unavailability problem is not solved after sharing all cached chunks and thus MSC between two swarms is meaningless).

4. Performance Evaluation

We use ns-2 simulator [13]. Each simulation includes two swarms with the same number of peers and the same content (256 KB chunk) to be shared. In the simulation, we focus on the performance study of various parameters including swarm size, $D$ and $\beta$ of our approach. Each swarm has one initial seeder and the seeder leaves its swarm after uploading whole chunk set. The leechers have 1200 Kbps download and 400 Kbps upload capacity and the seeder has 1200 Kbps upload capacity. Peers join the swarm with Poisson arrival pattern and leave the swarm after completing their downloads (unless they receive the cached chunk). The peers report the bitfield and the tracker monitors swarm status every $T$ seconds. For each simulation, we use 5 parameters including swarm size (default is 250), content size in MB (50), $D$
Fig. 2  Download completion ratio with different swarm sizes.

(a) BitTorrent

(b) MSC

Fig. 3  Download completion ratio with different number of cached chunks.

(a) D

(b) β

(0.9), β (0.99), and \( T \) in sec (30) to examine various aspects of MSC. For performance comparison, we implement normal BitTorrent (BitTorrent), existing bundling approach [5] (Bundling), and MSC-enabled BitTorrent (MSC). We implement Bundling by merging two swarms (two separate contents) into one swarm (content) with two seeders. We run each simulation 10 times and show the average across the results.

4.1 Performance Improvement

First, we examine a download completion ratio that is a ratio of a number of download completions to a number of participating leechers (Fig. 2). BitTorrent shows lower download completion ratio with larger swarm size (Fig. 2 (a)). In BitTorrent case, the leechers leave the swarm when they complete content downloads before some chunks can be replicated by other leechers. On the other hand, MSC shows much improved performance by at least 53% and at most 557% compared to BitTorrent (Fig. 2 (b)). MSC shows high download completion ratio (i.e., over 80% in most cases). This result shows that MSC is enough to improve the download completion ratio by providing the missing chunks through the caching and sharing of the rare chunks.

Figure 3 shows how the number of cached chunks affects the download completion ratio. It is worthwhile to re-mind that \( D \) and \( β \) affect the number of rare chunks to be cached in common but in a different way while they are totally independent. Basically, the lower \( D \) and \( β \), the more number of the chunks to be cached. Please note that \( β \) determines which peer will be included in calculating the content availability of swarm and \( D \) determines when a swarm needs to start MSC based on a seeder/leecher ratio. First observation from the results is that more number of cached chunks usually leads to better performance improvement. Actually, it is not easy to predict the missing chunks that will be unavailable after the seeder leaving. Thus, caching more number of chunks means higher probability of caching of the actual missing chunks. Second observation is, however, that the more number of cached chunks does not lead to linear increase of the performance. Actually, the performance can be improved regardless of the number of cached chunks as long as part or all of the actual missing chunks are cached while estimating the actual missing chunks is not easy.

Comparison results with Bundling are depicted in Fig. 4. Bundling increases the number of download completions by at least 11% and at most 266% compared to BitTorrent, but it is still less than MSC (Fig. 4 (a)). Specifically, MSC shows at least 26% and at most 334% increased number of download completions compared to Bundling. MSC also shows around 30% reduced download completion time compared to Bundling (Fig. 4 (b)). These results show that MSC is more effective to improve the number of download completions and the download completion time than Bundling in 1:1 multi-swarm collaboration.

We divide the leechers into peers who participate in MSC by caching the chunks (\( P \)) and others (\( N \)) to examine how MSC affects download completion time of the leechers. Figure 5 (a) shows an average download completion time of \( P \) and \( N \) with various swarm sizes. \( P \) shows reduced download completion time compared to \( N \), since \( P \) is supposed

\[ ^{1}\text{MSC and Bundling may show different results with different settings such as a number of seeders and a number of contents to be bundled.} \]
to receive the cached chunks first when the content unavailability happens although the difference between $P$ and $N$ is not so much (i.e., around 50 seconds). Please note that the difference can not be much, since the leechers upload the received chunks to at most two other leechers when they receive the cached chunks. This result verifies that our approach enables the missing chunks to be replicated by many leechers quickly. Figure 5 (b) shows the standard deviation of download completion time and this supports that most $P$ completes its content download with less deviation than $N$. In all cases, each $P$ caches one chunk (i.e., 0.5% storage overhead compared to the content it is downloading) and less than 10% of $P$ shares the cached chunks once during one interval (i.e., 256 KB/30 sec = 68.2 Kbps for chunk sharing across swarms), which means low overhead is given to $P$.

Now, we examine how the sharing of cached chunks affects the content availability before the content unavailability happens. MSC begins to share the cached chunks when one of the collaborating swarms faces the content unavailability. Thus, another swarm can utilize the cached chunks for improving chunk replication, even though it does not face the content unavailability. Figure 6 shows the content available period (i.e., a period from the seeder leaving to the happening of content unavailability). $R$ is the swarm that faces the content unavailability and $C$ is the counterpart swarm of $R$. In most cases, $C$ shows longer content available period than $R$. This result shows that the sharing of cached chunks can help the peers to replicate chunks sufficiently so that $C$ can enjoy better chunk replication before facing the content unavailability.

In summary, MSC with the current rare chunk estimation scheme is enough to increase the number of download completions compared to BitTorrent. Compared to Bundling under specific environment, MSC improves the number of download completions and the download completion time. $P$ can enjoy better performance than $N$ with low overhead, which can be an incentive for the peer participation. MSC is enough to improve the performance of the collaborating swarms regardless of the content unavailability.

4.2 Overhead of Caching and Sharing

MSC overhead including caching and sharing is shown in Fig. 7†. Figure 7 (a) shows that the caching overhead (i.e., the percentage of leechers caching chunks) drops significantly as the swarm size increases. This is due to that the number of cached chunks is not largely affected by the

†Due to the space limitation, we do not discuss the signaling overhead and overhead of the tracker.
swarm size and the caching overhead is distributed over leechers well as the swarm size increases. Specifically, the highest (lowest) caching overhead is 22.2 (1.93)%. The number of replications of shared chunks (in the swarm facing the content unavailability) increases substantially as the swarm size increases (Fig. 7 (b)) while the number of shared chunks across swarms is small (Fig. 7 (a)). It means that sharing chunks across swarms does not burden swarming performance of swarm sending the cached chunks and that the shared chunks are effectively replicated by leechers in the swarm facing the content unavailability when the cached chunks are shared across swarms. Above results show that MSC overhead for caching and sharing is low, but it has significant effect on performance improvement. Figure 7 (c) shows that the caching overhead is largely affected by $\beta$ and Fig. 7 (d) shows that the caching overhead increases gradually as the content size increases. In Fig. 7 (c) and Fig. 7 (d), **Additional downloads** means the performance improvement in terms of number of download completions by using MSC compared to normal BitTorrent case. Additional observation from Fig. 7 (c) and Fig. 7 (d) is that higher caching overhead does not lead to a corresponding linear increase of the performance although higher caching overhead usually results in better performance (as discussed earlier).

Compared to Bundling case, MSC shows similar or lower overhead in terms of storage and upload/download bandwidth consumption while showing negligible additional overheads. In Bundling case, if a peer wants to download all bundled contents, there will be no overhead. However, if the peer does not want to download some contents among the bundled contents, there will be some overheads as follows. Firstly, the peer has to spend more time in uploading and downloading the unwanted contents, which means more upload/download bandwidth is used. In addition, the download completion time will be increased. Another overhead is storage overhead. Until completing the download of all bundled contents, the peer has to use some storage space to store the unwanted contents. On the other hand, in MSC, the download completion time is not affected by MSC, since the peer is only utilized when the peer is online. Just part of peers is utilized for temporal caching (at most 22.2% in our simulation) and the peers selected for the caching need to cache one segment. We believe that the storage overhead for one segment is negligible compared to the content being downloaded (e.g., 0.5% storage overhead compared to the content it is downloading in our simulation). Even though MSC has additional overheads compared to Bundling, the additional overheads are negligible. Each peer reports its bitfield information together with Tracker Request and the size of bitfield is small. For example, in case of 1 GBytes content consisting of 4096 256 KB-sized chunks, the size of bitfield is 4906 bits = 4 Kbits. To match two collaborating swarms for MSC, the tracker just compares the number of rare chunks of candidate swarms, which requires simple comparison operations.

In summary, MSC can achieve significant performance improvement with low overhead. We anticipate that the caching overhead can be reduced without sacrificing the performance improvement if we can estimate the actual missing chunks correctly.

5. Conclusion

To improve the content availability of BitTorrent, we introduce a new kind of multi-swarm collaboration. In our approach, the swarm is regarded as temporal storage and thus
our approach enables any swarms to collaborate with each other by caching some chunks and by sharing the cached chunks at the proper time. Evaluation results show that the number of download completions can be much increased with low overhead compared to normal BitTorrent and the existing bundling approach. The results also show that our approach is enough to provide improved performance to the cooperative peers as a potential peer incentive and to provide better chunk replication to the counterpart swarm so as to suppress content unavailability. We anticipate that the performance improvement can be achieved with less overhead if we find a better way for rare chunk estimation and efficient chunk caching. Currently, we are working on this issue. Together with this, we plan to study M:M multi-swarm collaboration for additional performance improvement.

References

[1] B. Choen, “Incentives build robustness in BitTorrent,” Proc. Workshop on Economics of Peer-to-Peer Systems (P2PEcon), 2003.
[2] R. Macmanus, Trend watch: P2p traffic much bigger than web traffic. http://www.readwriteweb.com/archives/p2p_growth_trend_watch.php/, Dec. 2006.
[3] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, “A performance study of bittorrent-like peer-to-peer systems,” IEEE J. Sel. Areas Commun., vol.25, no.1, pp.155–169, Jan. 2007.
[4] Y. Yang, A.L.H. Chow, and L. Golubchik, “Multi-torrent: A performance study,” Proc. IEEE/ACM MASCOTS, 2008.
[5] D.S. Menasche, A.A.A. Rocha, B. Li, D. Towsley, and A. Venkataramani, “Content availability and bundling in swarming systems,” Proc. ACM CoNEXT, 2009.
[6] G. Dan and N. Carlsson, “Dynamic swarm management for improved bittorrent performance,” Proc. IPTPS, 2009.
[7] A. Ramachandran, A. das Sara, and N. Feamster, “Bitstore: An incentive compatible solution for blocked downloads in BitTorrent,” Proc. NetEcon, 2007.
[8] N. Carlsson and D.L. Eager, “Modeling priority-based incentive policies for peer-assisted content delivery systems,” Proc. IFIP Networking, 2008.
[9] M. Paitek, T. Isdal, A. Krishnamurthy, and T. Andserson, “One hop reputations for peer to peer file sharing workloads,” Proc. NSDI, 2008.
[10] S. Kaune, R.C. Rumin, G. Tyson, A. Mauth, C. Guerrero, and R. Steinmetz, “Unravelling bittorrent’s file unavailability: Measurements and analysis,” Proc. IEEE P2P, 2010.
[11] I. Al-Oqily and A. Karmouch, “Towards automating overlay network management,” Elsevier Journal of Network and Computer Applications, vol.32, no.2, pp.461–473, 2009.
[12] M. Yoshida and A. Nakao, “A resource-efficient method for crawling swarm information in multiple bittorrent networks,” Proc. International Workshops on Ad Hoc, Sensor and P2P Networks, 2011.
[13] Network Simulator ns-2, http://www.isi.edu/nsnam/ns/.

HyunYong Lee received the M.S. degree and the Ph.D. degree in computer science from the Gwangju Institute of Science and Technology (GIST), Korea, in 2003 and 2010, respectively. He is currently the researcher of the new generation network research center at the National Institute of Information and Communications Technology (NICT), Japan. His research interests include P2P traffic control and P2P content distribution.

Masahiro Yoshida received B.S. degree (2008) and the M.S. degree (2010) in Engineering from Tokyo University of Agriculture and Technology, Japan. He is currently the Ph.D. candidate of the University of Tokyo, Japan. He has also been a research fellowship for young scientists at Japan Society for the Promotion of Science (JSPS) since 2011. His research interests include P2P traffic control and network visualization.

Akihiro Nakao received B.S. (1991) in Physics, M.E. (1994) in Information Engineering from the University of Tokyo. He was at IBM Yamato Laboratory/at Tokyo Research Laboratory/at IBM Texas Austin from 1994 till 2005. He received M.S. (2001) and Ph.D. (2005) in Computer Science from Princeton University. He has been teaching as an Associate Professor in Applied Computer Science, at Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies, the University of Tokyo since 2005. (He has also been an expert visiting scholar/a project leader at National Institute of Information and Communications Technology (NICT) since 2007.).