Multi-Swarm Collaboration for Improving Content Availability in Swarming Systems

HyunYong Lee
National Institute of Information and Communications Technology (NiCT)
Tokyo, Japan
ifjesus7@gmail.com

Masahiro Yoshida and Akihiro Nakao
The University of Tokyo
Tokyo, Japan
yoshida@nakao-lab.org, nakao@iii.u-tokyo.ac.jp

***Abstract*—Despite its great success, BitTorrent suffers from a content unavailability problem where peers can not complete their content downloads due to some missing chunks, which is caused by an absence of seeders. Multi-swarm collaboration approach is a natural choice for improving the content availability, since the content unavailability can not be managed by one swarm easily. Most existing multi-swarm collaboration approaches, however, show 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 when the seeders are online 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 improves the number of download completions by over 50% compared to vanilla BitTorrent with low caching overhead. The results also show that our approach enables the peers participating in our approach to enjoy better performance than other peers, which can be an peer incentive.**

I. 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 a content unavailability where the peers can not construct a content with chunks available in a swarm. The content unavailability problem that is caused by an absence of seeders who have whole content 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 more than 50% of its swarming period [5].

Several existing work exploits a multi-swarm collaboration approach to increase the content availability, since the content unavailability can not be managed by one swarm easily. Although existing work improves the content availability, they show some content-related limitations, that 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 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 and requires an appropriate number of contents to be bundled for improving performance. The peers may also need to download content that they do not want to download content that they want.

In this paper, we try to improve the content availability based on the multi-swarm collaboration from 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 in near future when seeders are online and share the cached chunks when the content unavailability happens. In 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 measurement 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 the content unavailability. We perform simulations to examine our multi-swarm collaboration approach including performance improvement and corresponding overhead. Through simulations, we find that a number of download completions can be increased by over 50% with low caching overhead compared to vanilla BitTorrent (e.g., the highest caching overhead is 0.22 chunks per peer). 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. This observation leads us to study better

978-1-4577-0638-7 /11/$26.00 ©2011 IEEE

way for selection of chunks to be cached and efficient chunk caching as future work.

The remainder of the paper is organized as follows. Section II introduces our measurement result and related work. Section III discusses the proposed multi-swarm collaboration scheme in detail. Section IV evaluates our approach and Section V concludes this paper.

II. Content Unavailability and Multi-swarm Collaboration

A. Content Unavailability

In BitTorrent, content is divided into segments, called chunks. Each swarm consists of peers including leechers that have partial chunks and seeders that possess all chunks. The peers exchange chunks with each other using a tit-for-tat strategy 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 the content unavailability problem happens and this is mostly caused by an absence of the 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 finishing their downloads [10].

We conduct measurement including 1.5 million swarms to understand the content unavailability. Here, we show main observations (Table. I and Fig. 1). Detailed method for 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 50 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.

TABLE I
Distribution of number of seeders.

	0 seeder	1 seeder	>1 seeder
%	72.6	12.4	15

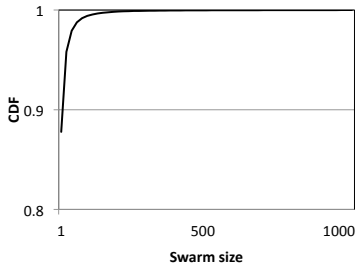


Fig. 1. Distribution of swarm size.

B. 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 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 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 increase the content availability and swarm lifetime 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 waiting time of peers in swarms with highly unavailable seeders. Dan et al. [6] shows how multiple swarms of 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 achieve much of its performance improvements and improved content availability.

On the other hand, rather than depending on the available contents (i.e., downloaded contents, contents to be bundled, and contents distributed through multiple swarms), this paper tries to improve the content availability without the content-related limitations by utilizing a swarm as a temporal storage.

III. Temporal Caching-based Multi-Swarm Collaboration

Our main goal is to improve the content availability by enabling the collaborating swarms to cache some chunks that the counterpart swarms may not be able to provide when the seeder goes offline. Thus, our main challenges include a time for the collaboration, selection of swarms to be matched for the collaboration, and selection of chunks to be cached. Corresponding state diagram is depicted in Fig. 2. From now on, for the sake of simplicity, we call our approach as MSC shortly. We assume that the original content publisher uploads at least one complete copy of content before leaving the swarm. In this paper, we define $A(S_i)$ as the content availability of swarm

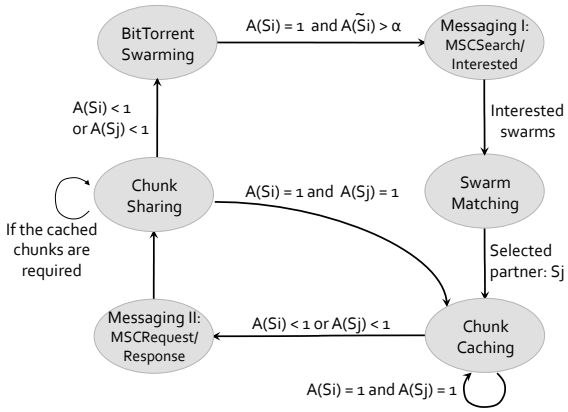


Fig. 2. State diagram of the proposed multi-swarm collaboration.

S_i, which is calculated as $\frac{\#\_of\_available\_chunks}{total\_number\_of\_chunks}$. We say that the content unavailability happens when $A(S_i) < 1$.

Tracker-based swarm monitoring (*BitTorrent Swarming*):[1] MSC utilizes the tracker, since the tracker is at the best position to understand overall status of the swarm. Basically, peers report a 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 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 first needs to select swarms to be considered for the collaboration. 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 a decision threshold: $A(S_i) = 1$ and $A(\tilde{S}_i) > \alpha$, where $A(\tilde{S}_i)$ is the content availability of S_i except the seeders and α is *min(1, D + seeder/leecer ratio)*.[2] $A(\tilde{S}_i)$ indicates an estimated content availability without the seeders. $A(\tilde{S}_i)$ increases when the seeder is still uploading the content or $A(S_i)$ is close to 1 when the seeder completes the content upload. Therefore, $A(\tilde{S}_i)$ may close to 1 as long as the seeders exist except initial content distribution period. On the other hand, α indicates a current seeding status and changes with the number of seeders. When there are enough seeders, the decision threshold cannot be met, since α may equal to 1 while $A(\tilde{S}_i)$ cannot over 1. If α decreases less than 1 due to few seeders, the decision threshold may be satisfied, since $A(\tilde{S}_i)$ may still close to 1. Thus, the decision threshold allows the swarms with few seeders to be selected for MSC.

Swarm matching (*Messaging I, Swarm Matching*): To match swarms for MSC, we utilize a number of chunks that are likely to be unavailable when the seeders and the leechers who have larger than $\beta\%$ of whole chunks leave the swarm (i.e., rare chunks). Here, we consider the leechers, since they may leave the swarm soon after completing the content download [10]. Detailed performance study about β affecting a number of rare chunks will be discussed later. Collaborating swarms cache same number of rare chunks of each other for fair collaboration. Even though the difference in the number of rare chunks can be covered by caching certain chunks multiple times, we believe that the more number of separate chunks to be cached, the better content availability, since it may cover more chunks. Thus, the swarm matching procedure is done as follows. When $A(S_i) = 1$ and $A(\tilde{S}_i) > \alpha$, a corresponding tracker, let say A sends *MSCSearch(e_i, S_i)* to other trackers after identifying the number of rare chunks of S_i, e_i. The 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 received *MSCInterested()* messages, the tracker A chooses one swarm that has smallest $|e_i - e_j|$. If $e_i > e_j$, S_i caches some rare chunks of S_j (e.g., chunks that have low replication ratio) multiple times so as to match total number of chunk cachings.

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 and let their peers who have the rare chunks send the rare chunks to the leechers of counterpart swarm. Each peer transfers at most one chunk to be cached for each interval not to degrade its swarming performance. For stable chunk caching, the leechers who have small number of chunks are selected to cache the rare chunks, since they are more likely to stay longer in a swarm than others until completing their downloads. If there are multiple leechers with same number of chunks, the tracker randomly selects the leecher. If the number of rare chunks is larger than swarm size, each leecher needs to cache one more chunks. In this case, for each rare chunk to be cached, the leecher who has smallest number of chunks among leechers caching smallest number of rare chunks is selected for the chunk caching.

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 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., $A(S_i) < 1$ or $A(S_j) < 1$). When $A(S_i) < 1$, the tracker A sends *MSCRequest(a list of missing chunks, leechers who will receive the cached chunks, S_i)* to the tracker B managing

[1]Italic word indicates the corresponding step of Fig. 2

[2]*D* is a system parameter and less than 1.

the counterpart swarm S_j. In return, the tracker B sends *MSCResponse()* including the same kind of information with *MSCRequest()*. In case of S_j that does not face the content unavailability, the missing chunks are chunks that have low replication ratio among the cached chunks. Then, the trackers let their leechers send the cached chunks once to the counterpart swarm so that the missing chunks are available. Basically, the leechers caching the 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. For better content replication, the leechers upload the received chunks to at most two leechers when they receive the cached chunks under the content unavailability.

Sharing chunks across swarms stops when the cached chunks are not required anymore and MSC re-starts from *Chunk Caching* (if $A(S_i)$ = 1 and $A(S_j)$ = 1) or *BitTorrent Swarming* (otherwise).

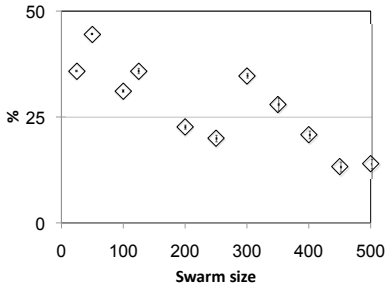
IV. Performance Evaluation

We use ns-2 simulator [13]. Each simulation includes two swarms with same number of peers and same content (256KB chunk) to be shared.[3] Each swarm has one initial seeder (that does not support a super seeding) and the seeder leaves the swarm after uploading whole chunk set. The leechers have 1200Kbps download and 400Kbps upload capacity and the seeders have 1200Kbps 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* (0.9), β (0.99), and *T* in sec (30) to examine various aspects of MSC. For performance comparison, we implement vanilla BitTorrent (BitTorrent) and MSC-enabled BitTorrent (MSC). We run each simulation 10 times and show the average across the results.

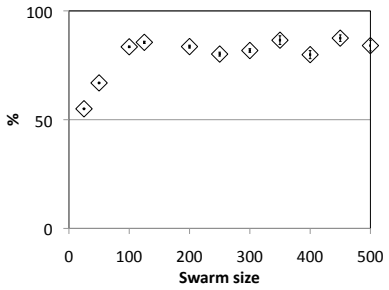
A. Performance Improvement

First, we examine a download completion ratio that is a ratio of a number of download completions to a number of participating leechers (Fig. 3). BitTorrent shows lower download completion ratio with larger swarm size (Fig. 3(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. 3(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.

[3]In this work, we focus on the performance study of various parameters including swarm size, *D* and β while leaving a study of caching redundancy for future work.



(a) BitTorrent

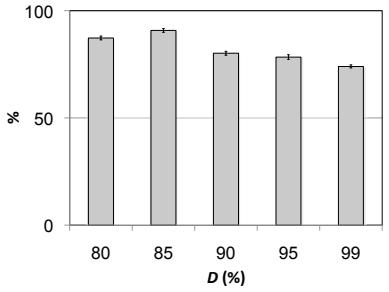


(b) MSC

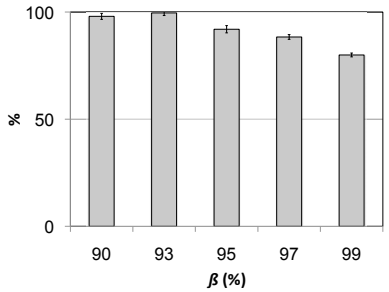
Fig. 3. Download completion ratio with different swarm size.

Fig. 4(a) and Fig. 4(b) show how the number of cached chunks affects the download completion ratio. It is worthwhile to note that *D* and β affect the number of rare chunks to be cached. For example, the lower *D* and β, the more number of the chunks to be cached. First observation 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. Even though the performance can be improved regardless of the number of cached chunks as long as partial or all of the actual missing chunks are cached, we cannot estimate the missing chunks accurately.

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. Fig. 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 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. Fig. 5(b) shows the standard deviation of download completion time and this

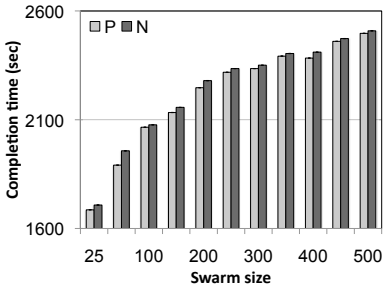


(a) D

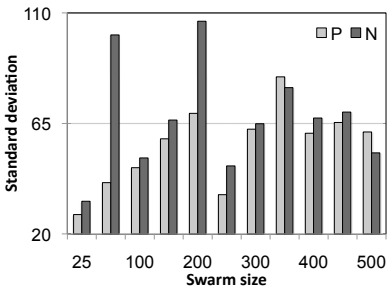


(b) β

Fig. 4. Download completion ratio with different number of cached chunks.



(a) Average



(b) Standard deviation

Fig. 5. Download completion time.

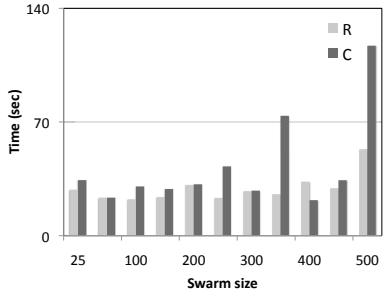


Fig. 6. Content available period.

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., 256KB/30sec = 68.2Kbps for chunk sharing across swarms), which means low overhead is given to P.

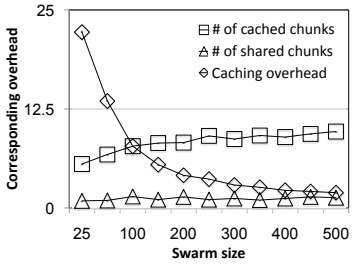
Now, we examine how the sharing of the cached chunks affects the content availability before the content unavailability happens. MSC begins to share the cached chunks when one of the collaborating swarms faces the content unavailability. Thus, another swarm can utilize the cached chunks, even though it does not face the content unavailability. Fig. 6 shows the content available period (i.e., from the seeder leaving to the 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 improve the number of download completions compared to BitTorrent. P can enjoy better performance with low overhead than N, 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. The results also lead us to find better rare chunk estimation scheme and efficient way to cache more chunks for additional performance gain as future work.

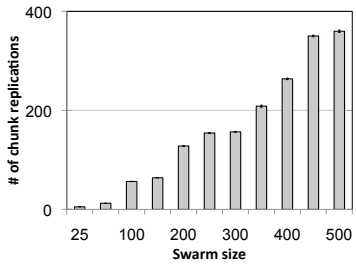
B. Overhead of Caching and Sharing

MSC overhead including caching and sharing is shown in Fig. 7.[4] Fig. 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 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

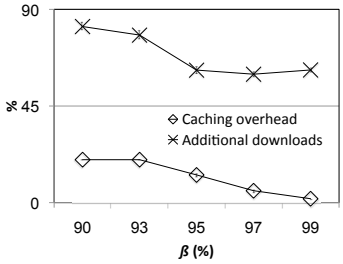
[4]Due to the space limitation, we do not discuss the signaling overhead and overhead of the tracker.



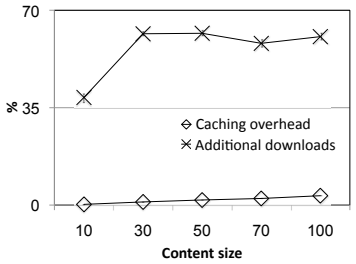
(a) Caching and sharing



(b) # of replications of shared chunks



(c) Various β



(d) Various content size

Fig. 7. MSC overhead.

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. Fig. 7(c) shows that the caching overhead is largely affected by β and Fig. 7(d) shows that the caching overhead increases gradually as the content size increases. 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).

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

V. Conclusion

Despite its great success, BitTorrent suffers from the content unavailability. To improve the content availability while overcoming the identified limitations of existing approaches, in this paper, we introduce a new kind of multi-swarm collaboration without content-related limitations. In our work, the swarm is regarded as a temporal storage and this approach enables any swarms to collaborate with each other by caching some chunks and by sharing the cached chunks at proper time. Evaluation results show that the number of download completions can be much improved with low overhead compared to vanilla BitTorrent. 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. 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," in *Proc. Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2003.

[2] Richard Macmanus, Trend watch: P2p traffic much bigger than web traffic. http://www.readwriteweb.com/archives/p2p_growth_trend_watch.php/, December 2006.

[3] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of bittorrent-like systems," *IEEE Journal on Selected Areas in Communications*, Issue 25, Jan. 2007.

[4] Y. Yang, A. L. H. Chow, and L. Golubchik, "Multi-torrent: A performance study," in *Proc. of 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," in *Proc. of ACM CoNEXT*, 2009.

[6] G. Dan and N. Carlsson, "Dynamic swarm management for improved bittorrent performance," in *Proc. of IPTPS*, 2009.

[7] A. Ramachandran, A. das Sara, and N. Feamster, "Bitstore: An incentive compatible solution for blocked downloads in BitTorrent," in *Proc. of NetEcon*, 2007.

[8] N. Carlsson and D. L. Eager, "Modeling priority-based incentive policies for peer-assisted content delivery systems," in *Proc. of IFIP Networking*, 2008.

[9] M. Paitek, T. Isdal, A. Krishnamurthy, and T. Andserson, "One hop reputations for peer to peer file sharing workloads," in *Proc. of NSDI*, 2008.

[10] S. Kaune, R. C. Rumin, G. Tyson, A. Mauthe, C. Guerrero, and R. Steinmetz, "Unravelling bittorrent's file unavailability: Measurements and analysis," in *Proc. of IEEE P2P*, 2010.

[11] I.AI-Oqily and A. Karmouch, "Towards automating overlay network management," *Elsevier Journal of Network and Computer Applications*, vol. 32, issue 2, 2009.

[12] Masahiro Yoshida and Akihiro Nakao, "A resource-efficient method for crawling swarm information in multiple bittorrent networks," in *Proc. of International Workshops on Ad Hoc, Sensor and P2P Networks*, 2011.

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