Time-Critical Data Dissemination in Cooperative Peer-to-Peer Systems

Chi-Jen Wu*†, Cheng-Ying Li†, Kai-Hsiang Yang†, Jan-Ming Ho† and Ming-Syan Chen*†
*Department of Electrical Engineering, National Taiwan University, Taiwan
†Institute of Information Science, Academia Sinica, Taiwan
{cjwu,cyli,hoho,khyang,mschen}@iis.sinica.edu.tw

***Abstract*—How to rapidly disseminate a large-sized file to many recipients is a fundamental problem in many applications, such as updating software patches and distributing large scientific data sets. In this paper, we present the Bee protocol, which is a cooperative peer-to-peer data dissemination protocol aiming at minimizing the maximum dissemination time for all peers to obtain time-critical data, such as critical patch updates. Bee is a decentralized protocol that organizes peers into a randomized mesh-based overlay and each peer only works with local knowledge. We devise a slowest peer first strategy to boost the speed of dissemination, and a topology adaptation algorithm that provides the most efficient utilization of the network capacity. Bee is designed to support network heterogeneity and deal with the flash crowd arrival pattern without sacrificing the dissemination speed. We present experimental results on the performance of Bee in terms of dissemination time and show that its performance can approach lower bound of the maximum dissemination time.**

I. Introduction

How to rapidly and efficiently distribute a large file across the Internet has become an interesting problem in the peer-to-peer (P2P) research community and some real applications, such as updating the software patches of Massively Multiplayer Online Games (MMOG) and operation systems in a flash crowd arrival pattern. Suppose that a large-sized critical data is initially held at a single server and we have to disseminate it to other N peers rapidly, how can we minimize the time it takes for all peers to have the complete data? This problem is very practical for the time-critical applications in the Internet.

Currently, a number of efforts focus on building P2P content delivery systems [1]–[4] to address the data dissemination problem. BitTorrent [1] is one of the pioneers of the file dissemination systems, and has become a prominent Internet application, both in terms of user popularity and traffic volumes [5]. BitTorrent is designed to reduce the load from congested servers and improve the download time of software. However, in such emergency conditions, such as dissemination of the critical patch updates for security or deployment of scientific data between institutions, e.g. CERN, all participants have to obtain data fast. For this reason, all participants should be cooperative for the fastest dissemination, the fairness mechanism, such as tit-for-tat scheme, are often unnecessary.

In this paper, we are interested in the **time-critical** data dissemination problem in which the maximum dissemination time of all the peers instead of dissemination time of individual peers is minimized. We begin by giving a formal definition of the data dissemination problem, and introduce the lower bound of the maximum dissemination time. We then present a decentralized protocol, called Bee, to address the time-critical data dissemination problem. Bee is a *best-effort* and and cooperative designed protocol to increase throughput of the system and individual peer. In the Bee protocol, peers are self-organizing into a random mesh and download blocks from neighboring peers. Moreover, Bee peer leverages a slowest peer first strategy and a topology adaptation algorithm to maximize the speed of dissemination. Under the slowest peer first strategy, a peer always transmits blocks to the neighbors that have the fewest number of downloaded blocks, i.e., the slowest neighbors. To decide the number of connections of a peer, Bee protocol embeds a topology adaptation algorithm based on peer's upload bandwidth. Our experimental results show that our Bee protocol can approach the lower bound of the data dissemination problem for both homogenous and heterogeneous network environments. Specifically, we make the following contributions in this paper:

We introduced the lower bound of the dissemination time for the data dissemination problem. And we proposed a P2P data dissemination protocol to minimize the data dissemination time for the time-critical applications. Although the authors [6] studied this problem, they focus on the theory deduction, we differ from them on analyzing how to design a protocol to approach the lower bound. The experimental results show that the new design can significantly reduce the overall download time and is scalable to a large number of users. Bee is very suitable for using to distribute the time-critical or emergency data to end users.

The rest of the paper is organized as follows. In next section, we discuss related work. In Section III, we first define the data dissemination problem. Section IV describes an overview and design details of the Bee protocol. Section V explains our simulation methodology and presents the performance results of the simulation study. We conclude this paper with a summary of main research results in Section VI.

II. Related Work

In recent years, there is a tremendous interest in building content delivery networks to address the data dissemination problem, which aims to deliver a large-sized file to a group of nodes spread across a wide-area network. For the content delivery networks, several related studies, such as Bullet [3],

Splitstream [7], and [8] build multiple overlay mulitcast trees or overlay meshes to establish a more efficient system to deliver data in the Internet. Though creating and maintaining multicast trees among peers provides an alternative solution for content delivery, however, it may incur high maintenance cost due to the characteristics of churn effect in a P2P network.

The popular content distribution system, BitTorrent [1] is successful for its efficiency in delivering a large file. There are two key mechanisms used in the BitTorrent system, namely, the "Tit-For-Tat" (TFT) peer selection policy and the local rarest first piece selection (LRF) strategy. Many varieties of BitTorrent are proposed to improve its performance. Wu *et al.* [9] try to reduce the download time of overall BitTorrent peers for flash crowd arrival pattern. In additional, Slurpie [2] focuses on reducing load on servers and clients download times when the number of downloading clients is huge. Crew [4] is a new gossip-based protocol for data dissemination, and it performs a faster dissemination than the BitTorrent protocol in experiments, and [10] is another similar work. Compared with the above efforts, our work makes a difference contribution on minimizing data dissemination time.

Content Delivery Networks (CDN), such as Akamai Technologies [11], have been proposed to improve accessibility for the commercial companies. CDNs are dedicated collections of servers located strategically across the wide-area Internet. Content providers, such as multimedia video sources, contract with commercial CDNs to distribute content. CDNs are compelling to content providers because the responsibility for distributing content is offloaded to the CDN infrastructure. Many new infrastructures for the CDNs have recently been developed to focus on distributing large files, such as CoBlitz [12] system. These systems offer a stable and performance predictable content delivery architecture, especially for the businesses that want to offload their bandwidth but need to delivery a large content. However, regardless of how many nodes in the CDNs are deployed, in case number of users grows too fast, performance of these systems may degrade significantly.

III. Data Dissemination Problem

In this section, we formally define the data dissemination problem, and provide the lower bound of the problem. Let us consider the problem of disseminating a file F to a set of n peers, $\mathcal{N} = \{1, 2, \ldots, n\}$. We also assume that a peer leaves the system once completely receiving the file. Let S be the server (which is called a "seed" in the rest of this paper) that has the file F in the beginning, and let $Size(F)$ denote the size of file F in bytes. Each peer $i \in \mathcal{N}$ in this system has its upload capacity U_i and download capacity D_i. And U_s represents the upper bound of the upload bandwidth utilization of seed S. We also assume that $U_i \leq D_i$, to model the state-of-the-art Internet technologies, such as ADSL or Cable modems. Due to the asymmetric nature of these network technologies, D_i is usually 3 to 5 times higher than U_i in practice. Let $t_i(F)$ denote the time it takes for peer i to receive the complete file F. Note that $t_i(F)$ denotes the time interval starting at the time peer i sends its request to the server and ending at the time it receives the entire file F. Before formally defining the problem, we define the following two performance metrics first.

*Definition 1 (**Average Dissemination Time**):*

$$ADT(F) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} t_i(F).$$

*Definition 2 (**Maximum Dissemination Time**):*

$$MDT(F) = \max\{t_i(F)\}, i \in \mathcal{N}.$$

Assume that the server and all n peers exist in the system from time $t = 0$. Then $MDT(F)$ is the time it takes for all peers to finish receiving the complete file F. Now we define the data dissemination problem as follows.

*Definition 3 (**Data Dissemination Problem**):* Given a server and n peers in a system, and each peer i has the upload capacity U_i and download capacity D_i, where $i = \{1, 2, \ldots, n\}$, the problem is to find a transmission scheme $\mathcal{M}$ to minimize the $MDT(F)$. According the definition 2, the problem can be formulated as a min-max problem as follows.

$$\min\{MDT(F)\}. \tag{1}$$

A. Ideal Dissemination Time (Lower Bound)

In this section, we focus on studying the lower bound of the dissemination time, which is also referred to as *ideal dissemination time* in this paper. Let us denote the actual amount of data uploaded by peer i as f_i, where $f_i \leq U_i \times t_i(F)$ and those peer i receives in return as r_i, where $r_i \leq D_i \times t_i(F)$. Without loss of generality, we assume that, the total amount of download data is equal to the total amount of upload data for the seed and all peers. Hence, we have the following equation.

$$f_s + \sum_{i=1}^{n} f_i = \sum_{i=1}^{n} r_i. \tag{2}$$

Since we are interested in estimating the lower bound of the dissemination time, we assume that upload capacity of each peer i is assumed to be fully utilized, i.e., we have $f_i = U_i \times t_i(F)$. Besides, the total amount of download bandwidth must be equal to $n \times Size(F)$, because all peers have the entire file F at the end. Then, we can extend the Eq. (2) as follows to deal with the ideal dissemination time for the general case.

$$U_s \times t_s(F) + \sum_{i=1}^{n} U_i \times t_i(F) = n \times Size(F). \tag{3}$$

Here, we have a min-max problem with its objective function in Eq. (1) subject to the constraints given by Eq. (3). Since the constraint is a linear equation, or more specifically, a hyperplane in $(n+1)$-dimension, and the optimal solution for the constrained optimization problem can be obtained if and only if when all t_i are the same, i.e.,

$$t_s(F) = t_1(F) = t_2(F) = \cdots = t_n(F). \tag{4}$$

Applying this results to Eq. (3), we then have a lower bound of $MDT(F)$, denoted by $\overline{T}(F)$, for file F, as follows.

$$\overline{T}(F) = \frac{n \times Size(F)}{U_s + \sum_{i=1}^{n} U_i}.$$

Thus, we have the following lemma.

LEMMA *1:* Let $T^*(F)$ denote the $MDT(F)$, of a feasible schedule of the data dissemination problem for a given file F. Then we have:

$$T^*(F) \geq \max\left\{\overline{T}(F), \frac{size(F)}{U_s}, \frac{size(F)}{\min\{D_i\}}\right\}, \qquad (5)$$

where the right-hand right of Eq. (5) is the lower bound of the dissemination time in any algorithm for the data dissemination problem.

IV. BEE DESIGN

In this section, we present the Bee protocol to approach the ideal dissemination time. In a Bee system, the content (file) is divided into many fix-sized blocks $B_i, i = \{1, 2, \ldots, m\}$, which is the smallest transfer unit in the system. We chose 256KB for the block size, which is the same as that used in most other P2P protocols. In the following, we start with an overview of the Bee protocol, followed by the detailed descriptions of its various components.

A. Overview of Bee

At a concept of overview, Bee constructs a random mesh overlay among a set of peers. Fig. 1 illustrates the scheme of a Bee system. Suppose that a large content is announced from a single server, and particularly we assume that $U_s > U_i$ in the system, and a lot of peers want to download the content at the same time. Note that it is reasonable, for example, an enterprise of online game, ex Blizzard needs to distribute the critical patch to end users. The download bandwidth of Blizzard's servers should be large than end-users. Each peer gets into contact with a centralized well-known register server and retrieves a *contact list* of an uniform random subsets of all peers in the system. The size of contact list is a small constant, say 160.

Based on the contact list, the peer discovers other peers, and exchanges update messages with them. The update message contains a *bitstring* about which blocks are available, and the bitstring can be used to coordinate the block requesting decisions without global information. After exchanging update messages, a peer could send requesting block messages to all peers in the contact list, and download blocks while uploading blocks it owns to other peers simultaneously. In a peer to peer network, peers may leave the system dynamically. In order to maintain connectivity of the overlay network, peers in Bee system will periodically, say 30 seconds, ask the register server to obtain a new contact list of replacement peers.

The key components in the Bee system include (1) a slowest peer first strategy for selecting peers to upload blocks, (2) a block selection strategy for requesting blocks, and (3) a topology adaptation algorithm for adapting the number of connections according to a peer's network capacity. The detailed descriptions of these components are presented as follows.

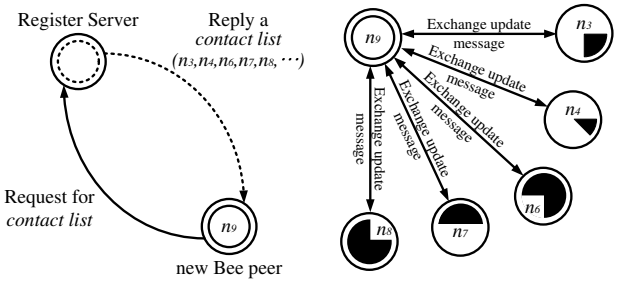


Fig. 1. A scheme of a Bee system.

B. Slowest Peer First Strategy

In this section, we describe the slowest peer first strategy. The design principle of the slowest peer first strategy is to keep all the upload capacity of peers full of data. This means that a peer can always find some peers to upload blocks to exhaust its upload capacity. Based on the slowest peer first strategy, a peer i always picks a slowest downloading peer among the contact list of the peer i, where the slowest downloading peer is the peer that has the least number of blocks. Consequently, the peer i can always upload blocks to the picked peer, because there is a high probability that the peer i has blocks that the picked peer does not have. From this point of view, Bee protocol adapts the slowest-peer-first strategy as a means to gain high utilization of the upload bandwidth of each peer.

After a peer joins into a Bee system, it periodically sends the requesting block messages to the peers in the *contact list* for downloading the blocks it lacks. When a peer starts to upload blocks to other peers, it maintains a *working set*, and tries to exhaust its network capacity to upload blocks to the peers in its working set. The working set consists of peers in the previous working set and those selected from the new contact list. The size of working set is adapted by a topology adaptation algorithm that we will discuss later.

The advantage of the slowest peer first strategy is to enable peers to diminish the MDT significantly. Disadvantage of the strategy is that it may increase the time it takes for the faster downloading peers to download the file. However, our goal is to minimize the MDT of the system. Note that this slowest peer first strategy dose **not** require all peers leaving at the same time, every peer can immediately depart the Bee system at an arbitrary time. We will show it in Section V.

C. Block Selection Strategy

Once a peer establishes connections with its neighboring peers, it needs to determine which blocks to request from which peers, based on the local knowledge (the available blocks in all peers among the contact list). The Bee protocol employs the local rarest first strategy for choosing new blocks to download from neighboring peers. The local rarest first strategy is proposed in BitTorrent protocol, and it can prevent the last block problem and increase the file availability in a BitTorrent system. The main advantage of the local rarest first

strategy is to overcome the last block problem by favoring rare blocks. This strategy equalizes the file block distribution to minimize the risk that some rare blocks are lost when peers owning them fail or depart the system. Bharambe *et al.* [13] study the local rarest first strategy by simulations and show that this strategy can address the last block problem efficiently. Another advantage of the local rarest first strategy is to increase the probability that a peer is useful to its neighboring peers because it owns the blocks that others do not have, and thus it helps to diversify the range of blocks in the system.

D. Topology Adaptation Algorithm

Bee is designed to adapt to different network environments by a topology adaptation algorithm. In general, the available bandwidth estimation is a non-trivial problem in practical network applications, so it is hard to decide how many upload connections a peer should have in the Bee protocol. For the sake of simplicity, we do not use the network bandwidth estimation techniques to determine the precise upload capacity of each peer. Instead, we assume that a user can input a coarse-grained bandwidth estimate, such as the form ADSL, Cable, T3, etc, that provides an initial maximum upload capacity estimate, U. In addition, we assume that peers (including the seed) have limited upload/download bandwidth but the Internet backbone is assumed to have infinite bandwidth. This assumption is reasonable, because the previous study [14] showed that the Internet backbone indeed has low utilization and The Internet's bottleneck almost always occurs at the last mile. Based on the two assumptions, we can develop a simple topology adaptation algorithm for the Bee protocol.

A simple adaptation approach is to set the upload rate for each upload connection to a same value, say rate α, for all peers. Hence, if a peer i has maximum upload capacity of U_i, it establishes $k = \lceil \frac{U_i}{\alpha} \rceil$ connections, where $\alpha \leq U_i$, $\forall\ i \in \mathcal{N}$. So in the Bee protocol, each peer establishes k concurrent upload connections. Note that k should be bounded by the size of the contact list. The idea is to identify capacity, i.e., k, of each peer so that number of peers served by a peer would match its own capacity. Each peer can fully utilize its upload capacity and maximize its contribution to the system throughput.

V. Performance Evaluation

To understand the performance of our protocol, we built a discrete-event simulator to simulate the distribution of a large file from a server to a large number of peers in the Bee protocol. Our simulator is based on the paper [13]. In our simulator, the network model associates a download link and an upload link bandwidth with each peer in order to make it suitable for modeling asymmetric access networks.

A. Road-map of Simulations

We made our simulations to compare the dissemination time in the Bee system with the lower bound of dissemination time and the required time in BitTorrent. Besides, we consider three network scenarios to evaluate our protocol, each representing a different degree of heterogeneity in their upload/download capacity and the peer join pattern is set to flash crowd.

TABLE I
THE UPLOAD/DOWNLOAD BANDWIDTH DISTRIBUTION

Network Type	Downloadlink	Uplink	Fraction
Homogeneous	1500kbps	384kbps	100%
Heterogeneous	1500kbps	384kbps	50%
	3000kbps	1000kbps	50%
More heterogeneous	784kbps	128kbps	20%
	1500kbps	384kbps	40%
	3000kbps	1000kbps	25%
	10000kbps	5000kbps	15%

The bandwidth distribution of each network condition is presented in Table I. Especially, the more heterogeneous condition with four types of peers is the actual peer bandwidth distribution which is reported from Gnutella clients [15]. We also present the results of a realistic join pattern that derived from a tracker log for a Redhat 9 distribution torrent of a BitTorrent system [13]. Unless otherwise specified, we use the following settings in our experiments. First, we used a file size of 200MB with a block size 256KB (so a file contains 800 blocks). The seed's upload capacity is 6000Kbps. The number of contact list is 80 in both the Bee and BitTorrent system. Then the number of initial seeds is only one in all of our experiments. In addition, based on our experiments [16], we choose the uploading rate $\alpha = 25$ Kbps in the following experiments.

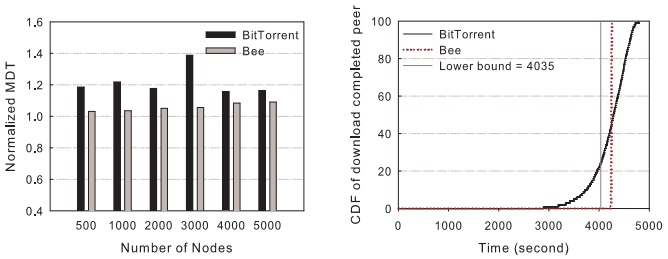
B. Homogeneous Environment

We start by comparing the performance of Bee with that of BitTorrent protocol in the homogeneous environment. We use the default settings as mentioned in the above description. All peers join the system at the initial stage, and leave the system when they finish their downloading.

First, we consider the impact of the scalability on the performance of Bee and BitTorrent. We use a different number of peers from 500 to 5000 in experiments, and all peers join system at the initial stage and remain in the system until they have a complete file. Fig. 2 (a) shows the normalized MDT. In our experiments, Bee always shows better performance in terms of MDT than BitTorrent regardless of network size. Moreover, the difference ratio between Bee and the lower bound is only 1.1 at a network with 5000 peers. Fig. 2 (b) shows the cumulative distribution of the number of complete peers in a network with 2000 peers. The results show that our Bee can efficiently diminish the maximum dissemination time, and the MDT of Bee is only 4218 seconds that is very close to the lower bound.

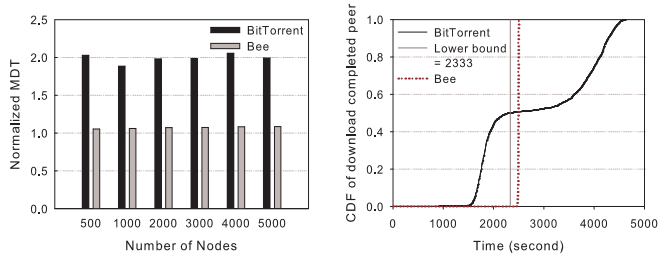
C. Heterogeneous Environment

Next, we evaluate the performance of Bee and BitTorrent protocol in a heterogeneous network that consists of two types



(a) MDT (b) CDF

Fig. 2. Scalability comparison in homogeneous environments.



(a) MDT (b) CDF

Fig. 3. Scalability comparison in heterogeneous environments.

peers, one of which has a higher download/upload capacity (3000/1000 Kbps) than the other (1500/384 Kbps). In this scenario, the lower bound is 2333 seconds. Fig. 3 (a) shows the normalized MDT metric for Bee and BitTorrent, and we can see that Bee is almost twice faster than the BitTorrent in the MDT metric. We also show the cumulative distribution of the number of complete peers in a network with 2000 peers in Fig. 3 (b). The result shows that a peer with higher capacity leaves faster than the peer with lower capacity in BitTorrent. After the peers with higher capacity leave system, the system capacity deceases significantly and the dissemination time of the peers with lower capacity will be prolonged.

D. More Heterogeneous Environment

In this section, we repeated the same experiment in a more heterogeneous network. Actually, it presents a very complex network condition. As previous experiments, we calculate the lower bound of this scenario and it should be 2089 seconds ($\frac{size(F)}{\min\{D_i\}} = \frac{1638400}{784}$). Our first experiment is to study the impact of the number of peers from 500 to 5000. We now explore the scalability of Bee in the complex network environment.

Fig. 4 (a) shows the normalized MDT metric of Bee and BitTorrent. This result also shows that Bee is almost twice faster than the BitTorrent in MDT metric. We also show the cumulative distribution of the number of complete peers in a network with 2000 peers in Fig. 4 (b). Here, the dashed vertical line in Fig.4(b) denotes the lower bound of the data dissemination time. It is easy to see that 80% peers leave system at the $\overline{T}$ time (Recall that the Eq. (5) that we defined in section II.) and the remaining 20% peers prolong the MDT of Bee system. In fact, these 80% peers are higher capacity peer, and the dissemination time of remained peers is limited by their download capacity. This result also shows that the slowest peer first strategy of Bee dose **not** enforce higher capacity peers leaving the system with all peers at the same time, every peer can immediately depart when it received complete file.

In order to demonstrate this phenomenon, we extend the download capacity of low-capacity peers (784 Kbps) to make sure that each peer can download the complete file before the lower bound. We increased the download capacity of the poor capacity peers from 784 Kbps to 1200 Kbps. Fig. 4 (c) shows the CDFs of the download time for the two protocols in the case of a network with 2000 peers. The top one is the CDF with 784 Kbps (the minimum download capacity of some peers) and the bottom is the CDF with 1200 Kbps. From the results in Fig. 4 (c), we can observe that the maximum dissemination time of Bee also approaches the lower bound, when the lower bound is dominated the factor max$\{$ $T(F)$, size(F)/U_s, size(F)/min(D_i)$\}$ instead of the factor size(F)/min(D_i). The result implies that the performance of Bee is independent of the degree of network heterogeneity, and the dissemination time of Bee can approach the lower bound when there are no bottleneck links at the downstream peers. In sum, all peers in Bee are able to finish downloading very quickly even in the complex network scenario.

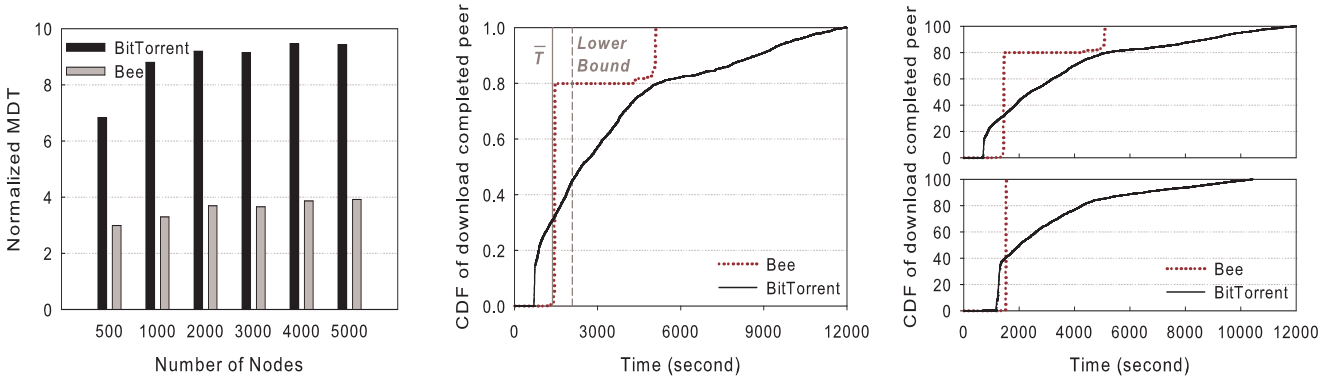
E. The Effect of Join Pattern

In this section, we study the impact of different user arrival patterns on the performance of Bee and BitTorrent systems. In following experiments, we vary the peer join rate to evaluate the performance variation of Bee and BitTorrent systems. All experiments in this section use the two following sets of settings: 1) a poisson arrival process with a total of 1000 peers. 2) an arrival pattern is derived from a tracker log of a Redhat 9 distribution torrent. The capacity of each peer is randomly selected among four capacity types as shown in Table I.

Fig. 5 shows the performance when using different user arrival rates for Bee and BitTorrent system. As Fig. 5 (a) shows, we can easily observe that when the arrival rate is low, the MDTs of Bee and BitTorrent both increase. We may observe that MDTs of Bee and BitTorrent at the arrival rate of 0.1 peers/sec are both higher than those at much higher arrival rates. The reason is that, when the arrival rate is low, the service capacity of overall system is also low. So the peers in our Bee or BitTorrent need more time to finish downloading the file. However, our Bee can outperform BitTorrent in the metrics in such scenario.

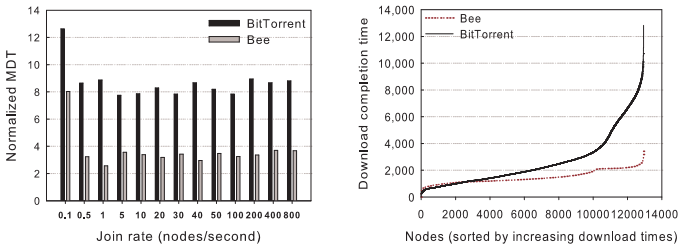
We now evaluate Bee and BitTorrent in a realistic join pattern. In this experiment, each peer joins the system according to the tacker log of a Redhat 9 distribution torrent, the log was collected over 12,000 peers joining time. Note that peer capacity consists of four types as shown in Table I, so the lower bound in this case is that $\frac{size(F)}{\min\{D_i\}} = 2089$ seconds.

We show the download completion time of each peer for Bee and BitTorrent in Fig. 5 (b). 83% peers in Bee finish the their download before 2000 seconds. On the contrary, only 50% peers can leave BitTorrent system at 2000 seconds.



(a) MDT (b) (c)

Fig. 4. Scalability comparison in the more heterogeneous environment.



(a) MDT (b) Distribution of dissemination time

Fig. 5. Performance comparison of Bee to BitTorrent with increasing arrival rate in more heterogeneous environment.

Moreover, Bee only needs 1/3 download time of BitTorrent to finish the file dissemination. The result shows that Bee can significantly reduce the overall download time of the file dissemination system, and this is a significant result as the dissemination time is a principal metric for time-critical applications.

We especially concern how much delay occurred in the dissemination time for those peers with higher upload bandwidth. In Fig. 5 (b), all the peers with higher upload bandwidth are the first 2,000 nodes; it is easy to see that the downloading times of the high-bandwidth peers only have a slight increase, which means the high-bandwidth peers only have a little delay in the Bee system. The result shows that our Bee protocol does **not** enforce the high-bandwidth peers to stay in the system.

VI. Conclusion

In this paper, we present our experimental study of the time-critical data dissemination problem. We present the Bee protocol for disseminating the time-critical data in the Internet. Bee protocol includes a slowest peer first strategy and a topology adaptation algorithm to maximize the speed of dissemination. Under the slowest peer first strategy, a peer always transmits blocks to the neighbors that have fewest number of downloaded blocks. Bee protocol also embeds a topology adaptation algorithm for a peer to adapt number of connections to its neighbors based on its own upload bandwidth. Moreover, our experimental results show that the dissemination time of the Bee protocol approaches lower bound of the data dissemination problem for both homogenous and heterogeneous network environments. Specifically, in the simulations on heterogeneous network environment presented in this paper. As for the arrival traffic derived from a software release log, Bee can significantly reduce the overall download time of the file dissemination system than the BitTorrent system.

References

[1] B. Cohen, "Incentives build robustness in bittorrent," *Proc. of ACM P2PECON*, 2003.
[2] R. Sherwood, R. Braud, and B. Bhattacharjee, "Slurpie: A cooperative bulk data transfer protocol," *Proc. of IEEE INFOCOM*, 2004.
[3] D. Kosti, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," *Proc. of ACM SOSP*, 2003.
[4] M. Deshpande, B. Xing, I. Lazardis, B. Hore, N. Venkatasubramanian, and S. Mehrotra, "Crew: A gossip-based flash-dissemination system," *Proc. of IEEE ICDCS*, 2006.
[5] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy, "Transport layer identification of p2p traffic," *Proc. of ACM IMC*, 2004.
[6] R. Kumar and K. Ross, "Peer assisted file distribution: The minimum distribution time," *Proc. of IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2006.
[7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," *Proc. of ACM SOSP*, 2003.
[8] X. Zheng, C. Cho, and Y. Xia, "Optimal peer-to-peer technique for massive content distribution," *Proc. of IEEE INFOCOM*, 2008.
[9] C.-J. Wu, C.-Y. Li, and J.-M. Ho, "Improving the download time of bittorrent-like systems," *Proc. of IEEE ICC*, 2007.
[10] A. Papadimitriou and A. Delis, "Flash data dissemination in unstructured peer-to-peer networks," *Proc. of the IEEE ICPP*, 2008.
[11] "Akamai technologies, inc." *http://www.akamai.com/*.
[12] K. Park and V. S. Pai, "Scale and performance in the coblitz large-file distribution service," *Proc. of USENIX NSDI*, 2006.
[13] A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving bittorrent performance," *Proc. of IEEE INFOCOM*, 2006.
[14] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," *Proc. of ACM IMC*, 2003.
[15] S. Saroiu, P. K. Gummadi, and S. D, "A measurement study of peer-to-peer file sharing systems," *Proc. of ACM MMCN*, 2002.
[16] C.-J. Wu, C.-Y. Li, K.-H. Yang, and J.-M. Ho, "Bee: A best effort peer-to-peer delivery protocol for internet data dissemination applications," *Technical Report (TR-IIS-06-015), IIS, Academia Sinica*, 2006.