IEEE Open Journal of the Computer Society

Received 6 January 2022; accepted 1 February 2022. Date of publication 8 February 2022;
date of current version 23 February 2022. The review of this paper was arranged by Associate Editor Zhaolong Ning.

Digital Object Identifier 10.1109/OJCS.2022.3149411

Time-Critical Data Dissemination Under Flash Crowd Traffic

CHI-JEN WU [1] (Member, IEEE), AND JAN-MING HO [2] (Senior Member, IEEE)

[1] Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan City 333, Taiwan
[2] Institute of Information Science, Academia Sinica, Taipei City 115, Taiwan

CORRESPONDING AUTHOR: CHI-JEN WU.

This work was supported by the Ministry of Science and Technology of Taiwan under Contract MOST110-2222-E-182-003-.

ABSTRACT How to rapidly disseminate a large-sized file to many recipients in flash crowd arrival patterns is a fundamental challenge in many applications, such as distributing multimedia content. To tackle this challenge, we present the Bee, which is a time-critical peer-to-peer data dissemination system aiming at minimizing the maximum dissemination time for all peers to obtain the complete file in flash crowd arrival patterns. Bee is a decentralized system that organizes peers into a randomized mesh-based overlay, and each peer only works with local knowledge. We introduce the slowest peer first strategy to boost the speed of dissemination and present a topology adaptation algorithm that adapts the number of connections based on upload bandwidth capacity of a peer. Bee is designed to support network heterogeneity and deals with the flash crowd arrival pattern without sacrificing the dissemination speed. We also show the lower bound analysis of the data dissemination problem, and present the experimental results to demonstrate that the performance of Bee can roughly approximate the lower bound of the data dissemination problem under flash crowd traffic.

INDEX TERMS Peer-to-peer, content distribution, flash crowd.

I. INTRODUCTION

How to rapidly distribute a large file in flash crowd arrival patterns [1] has become more and more attractive in the networking research community and mobile cloud computing applications, such as the emerging techniques for mobile content caching [2] and fast delivery [3] or updating the software patches of Massively Multiplayer Online Games (MMOG) [4] and operating systems [5]. Suppose that a large file is initially held by a single server, we have to disseminate it to other n peers, and it is the dissemination problem we defined: how to minimize the maximum dissemination time for all peers to obtain the complete file, especially in heterogeneous and dynamic networks? Furthermore, when a popular content is released, the peer arrival rate results in a flash crowd [1] as shown in Fig. 1. It should increase the difficulty of system design and significantly impact on the system performance.

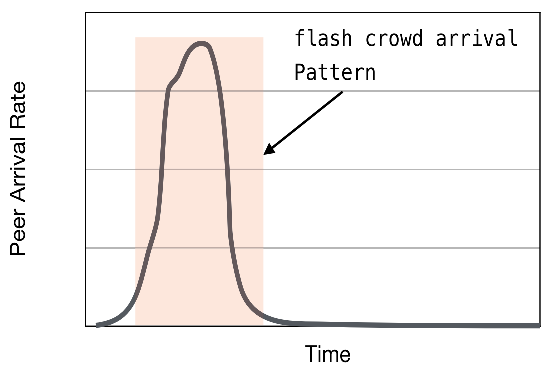


FIG. 1. An example of a flash crowd arrival pattern.

Under flash crowd traffic, several characteristics make it not easy to design a scalable system that organizes resources, such as computing power and network bandwidth, for disseminating content to a large number of clients, including: **1) Scalability:** the number of participating nodes must be in the thousands or even more. **2) Churn:** the behavior of participating users is characterized by the dynamics with which the peers join, leave, and rejoin the system at an arbitrary time, making it difficult to maintain an overlay network among a large number of participators. **3) Heterogeneity:** the resources, such as bandwidth, computing power of participating peers are heterogeneous, which make it difficult to make a schedule in polynomial time. **4) Network dynamic:** routers, links in the Internet, and the peers may fail, incurring more communication cost and longer transmission time to deliver content. Thus, it is not easy to maintain a large-scale data dissemination system that minimizes the maximum dissemination time under flash crowd traffic.

A famous peer-to-peer (P2P) content delivery protocol is the BitTorrent [6], which is one of the pioneers of content dissemination. However, BitTorrent is designed to minimize the dissemination time of **each peer** egoistically. On the other hand, previous studies, including Slurpie [7], Bullet [8], M2M-ALM [9] and ReCREW [10], O-Torrent [11], [12] and [13], have proposed to construct and maintain an overlay network of multiple trees, rings, or a random mesh to deliver content from a single server. In the design of these P2P protocols, content are usually divided into m parts of equal size, each being called a block. A peer may download any one of these blocks either from the server or from a peer who has downloaded that block. Besides, many researchers, including [14]–[18], have studied the performance of these P2P protocols, and have shown that the P2P protocol is both efficient and scalable, even it lacks a centralized coordination and scheduling mechanisms. Nevertheless, these previous studies related to content dissemination do not force on minimizing the maximum dissemination time for all requesting peers under flash crowd traffic.

In this article, we investigate the data dissemination problem that arises as an attractive application in the current Internet, such as content caching in [19], fast content delivery among mobile edge nodes [20], updating software patches or distributing multimedia content, especially in a flash crowd scenario. More specifically, we investigate and address the following two questions:

i) what is the lower bound of the data dissemination problem?

ii) how to design a distributed system that can achieve or approach to the lower bound of the data dissemination problem under flash crowd traffic?

We begin by giving a formal definition of the data dissemination problem. We then derive the lower bound of the ideal dissemination time both in homogeneous and heterogeneous networks.

Then we present Bee, a time-critical P2P data dissemination protocol to address the data dissemination problem under flash crowd traffic. Bee is designed from a *best-effort* service concept, to increase the system throughput and peer concurrency. Based on the *best-effort* service concept, a peer allocates uploading bandwidth for all its neighbors and attempts to serve all of them. Peers in Bee begin by self-organizing into a random overlay mesh and download blocks from their neighbors as soon as possible.

We present the slowest peer first strategy and the topology adaptation algorithm to maximize the speed of content dissemination. Based on the slowest peer first strategy, a peer transmits blocks to the higher priority neighbors that have the fewest number of downloaded blocks. The topology adaptation algorithm is for a peer to adapt the number of connections to neighbors based on its uploading capacity. Our experimental results demonstrate that the maximum dissemination time of the Bee can approximately approach the lower bound of the data dissemination problem. To the best of our knowledge, this work is the first that studies the effects of P2P protocols with respect to minimizing **maximum** data dissemination time under flash crowd traffic.

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

II. DATA DISSEMINATION PROBLEM

In this section, we formally define the data dissemination problem and show the lower bound of this problem.

Let us consider the problem of disseminating a file F to a set of n peers, $\mathcal{N} = \{1, 2, \ldots, n\}$. We assume that a peer leaves the system once completely receiving the file. Let S be the server (we called it *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. U_i and D_i respectively represent the upper bound of the upload and download bandwidth for peer i. We also assume that $U_i \leq D_i$, to model the Internet technologies. Let $t_i(F)$ denote the time it takes for peer i to download 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 data dissemination problem, we define the following two performance metrics first.

Definition 1 (Average Dissemination Time, ADT(F)):

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

Definition 2 (Maximum Dissemination Time, MDT(F)):

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

Assume that the server S 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 S and n peers in the system, and each peer i has the upload capacity U_i and download capacity

D_i, where $i = \{1, 2, \ldots, n\}$, the data dissemination problem is to find a transmission mechanism $\mathcal{M}$ to minimize the $MDT(F)$. According to the Definition 2, the data dissemination problem can be treated as a min-max problem as follows.

$$\min\{\max\{t_i(F)\}\}. \quad (1)$$

Note that the data dissemination problem is somewhat different from the broadcast network problem [21] where a node is required to broadcast a message to all other nodes as fast as possible. In the broadcast network problem, a node can either transmit a message to or receive a message from other nodes, but not both. Many researchers have studied the broadcast network problem in homogeneous networks for more than 40 years [21]. The broadcast network problem becomes NP-hard [22] in heterogeneous networks, and Deshpande *et al.* [23] proposed two centralized heuristics for the broadcast network problem in heterogeneous networks. In the data dissemination problem, however, a node can transmit and receive messages simultaneously. Goetzmann *et al.* [24] show that the data dissemination problem becomes NP-hard for equal upload and download capacities per peer. Thus, it also is more difficult to analyze algorithmic complexity of the data dissemination problem in heterogeneous networks.

A. IDEAL DISSEMINATION TIME (LOWER BOUND)

In this section, we focus on studying the lower bound of the data dissemination time, denoted as the **ideal dissemination time** in homogeneous or heterogeneous networks. We assume that peers are highly cooperative and altruistic, so that each peer is willing to forward data to other peers as fast as possible (*best-effort* service concept) and to benefit other peers than itself.

Let's denote the actual amount of data uploaded by peer i as f_i, where $f_i \leq U_i \times t_i(F)$ and the peer i receives in return as r_i, where $r_i \leq D_i \times t_i(F)$. Without loss of generality, we may assume that the total amount of download data must be equal to the total amount of upload data for the seed S and all peers. Hence, we have the following equation,

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

Since we are interested in estimating the lower bound of the data dissemination time, we assume that upload capacity of each peer i is 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,

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

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, a hyperplane in $(n+1)$-dimension, we show that an optimal solution for the constrained optimization problem can be obtained if and only if when all t_i are the same, i.e.,

$$t_s(F) = t_1(F) = t_2(F) = \cdots = t_n(F). \quad (4)$$

Lemma 1: Given a file F to a set of n peers, $\mathcal{N} = \{1, 2, \ldots, n\}$ and a seed S, the ideal dissemination time can be obtained if and only if when all t_i are the same.

Proof: We prove Lemma 1 by contradiction. We assumed there is an optimal solution α and $t_j^{\alpha}(F) < t_s^{\alpha}(F) = t_1^{\alpha}(F) = t_2^{\alpha}(F) = \cdots = t_n^{\alpha}(F)$, where $1 \leq j \leq n$. So,

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

Eq. 3 implies that the assumption is a contradiction. Thus, we have Lemma 1. ■

Applying Eq. 4 to Eq. 3, we then have a lower bound of $MDT(F)$, denoted by $\overline{T}(F)$, for file F, as follows.

$$\overline{T}(F) = \frac{n \times Size(F)}{U_s + \sum_{i=1}^{n} U_i}. \quad (5)$$

Now, we are ready to show the following Lemma 2.

Lemma 2: Let $T^*(F)$ denote the $MDT(F)$, of a feasible schedule of the data dissemination problem for a given file F. Then we have:

$$T^*(F) \geq \max\left\{\overline{T}(F), \frac{size(F)}{U_s}, \frac{size(F)}{\min\{D_i\}}\right\}, \quad (6)$$

where the right-hand right of Eq. 6 is the lower bound of the dissemination time in any algorithm for the data dissemination problem.

Proof: Note that the lemma merely says that $T^*(F)$ must be greater than 1) the lower bound (the ideal dissemination time), the Eq. 5 we derived in the above; 2) the time for the seed S to transmit the file F, the bottleneck is in the uplink capacity of the seed S; 3) the time for the slowest peer to download the file F, the bottleneck is in the download capacity of the slowest peer. ■

This analysis indicates that if someone has to design a time-critical data dissemination system to approach the ideal dissemination time in a general network environment, the two prerequisite concepts should be considered: 1) the system should enforce the peers to leave the system at almost the same time, and 2) the system should always fully utilize the upload capacity of each peer. Based on the two observations, we present our design of the Bee protocol in the next Section III.

III. BEE DESIGN

In this section, we present the Bee system to approach the ideal dissemination time that we derived in Section II. In the Bee, like other P2P content delivery systems [6], the file is divided into many fix-sized blocks (256 KB). The proposed Bee consists of 3 parts: slowest peer first strategy, local rarest first strategy, and topology adaptation algorithm. The slowest peer first strategy is used to control and balance the dissemination process of all nodes. By this strategy, nodes might have the same download process and complete the download at almost the same time with high probability. The local rarest first strategy can prevent the last block problem and increase blocks availability. The topology adaptation algorithm is used to dynamically adjust different uplink capacities and make nodes fully utilize the upload capacity with high probability.

At a high level overview, Bee organizes a random mesh overlay among a set of participating peers. Suppose that a large file is announced from a single seed S, then peers require to download the content at the same time. Each peer gets into contact with well-know register server and retrieves a *contact list* of a uniform random subsets of all peers. The size of contact list is a small constant, say 80. The initial mechanism is the same as that used in other P2P content delivery systems.

A. SLOWEST PEER FIRST STRATEGY

We describe the slowest peer first strategy and the procedure that each peer performs. Overall, a good peer selection strategy would be one which neither requests a peer to maintain a global knowledge, nor to communicate with many peers, but the one which is able to find peers having blocks the peer needs. In order to minimize the MDT of the dissemination system, our focus is the development of a distributed system, which each peer learns its nearby peers' statuses (with local knowledge) and selects a suitable peer to upload blocks.

The design principle of the slowest peer first strategy is to fully utilize all the upload capacity of peers in the system. It implies that a peer can always find some peers to upload blocks to utilize its upload capacity. Based on the slowest peer first strategy, a peer i always picks the slowest downloading peer among its *contact list*, 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 to utilize its upload capacity, because there is a high probability that the peer i has some blocks that the picked peer does not have. In this point of view, the slowest peer first selection strategy makes our Bee system to be able to maintain a high level of throughput of peer's upload capacity.

The operation of the slowest peer first strategy is efficient and sustainable for fully utilizing the upload capacity of each peer. Figure 2 illustrates the idea of the slowest peer first strategy. After a peer joins, 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 the peer tries to utilize its upload capacity as much as possible to upload blocks to the peers in its working set. The working set is a set of peers selected from the contact list over a period of time. The size of working set is controlled by the topology adaptation algorithm that we will discuss later. We present the pseudocode of the slowest peer first algorithm in Algorithm 1 as follows.

The advantage of the slowest peer first strategy is to enforce peers to download blocks at approximately the same progress and the design can significantly diminish the MDT of the system. The disadvantage is that the faster peers will be delayed by the slower peers. However, if the faster downloading peers leave the system early, the MDT of the system will be prolonged undoubtedly, recalling the Lemma 1 in Section II.

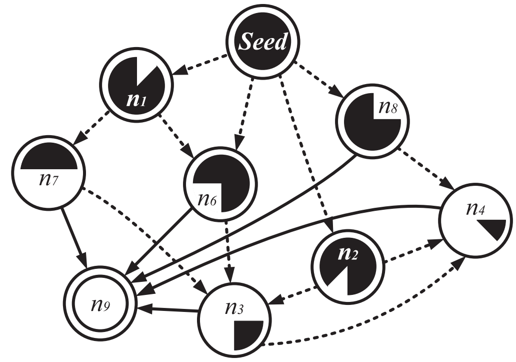


FIG. 2. An illustration of the slowest peer first strategy: The shaded content within a peer represents the percentage of the file that a peer has downloaded.

Algorithm 1: Slowest Peer First Strategy.

Begin
$WorkingSet[] \leftarrow Null$
for $j \leftarrow 0$ **to** $Sizeof(WorkingSet[])$ **do**
  Pick the slowest peer $i \in$ *Contact List*
  $WorkingSet[j] \leftarrow$ peer i
  $j \leftarrow j + 1$
end for
return $WorkingSet[]$
End

B. BLOCK SELECTION STRATEGY

Bee employs the local rarest first strategy for choosing new blocks to download from neighboring peers. The local rarest first strategy is proposed in BitTorrent [6], 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 [25] 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.* [26] 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. Thus, the local rarest first strategy helps diversify the range of blocks in the system.

C. TOPOLOGY ADAPTATION ALGORITHM

The design of Bee explicitly takes into account the capacity heterogeneity associated with each peer in P2P networks. Bee leverages the topology adaptation algorithm to dynamically adjust different uplink capacities. In general, the available bandwidth estimation [27] is a non-trivial problem, so it is hard to decide how many upload connections a peer should have in a Bee system. However, using a fixed number of upload connections will not perform well under a wide variety of peers' uplink capacities. Hence, Bee leverages a topology adaptation algorithm that attempts to dynamically maintain the maximum number of upload connections according to the upload capacity of each peer.

We do not use the network bandwidth estimation techniques [28] to determine the precise uplink capacity of each peer. Instead, we assume that the user can configure a coarse-grained bandwidth that provides an initial maximum upload capacity U_i. In addition, we assume that peers (including the seed S) have limited upload/download bandwidth but the Internet backbone has infinite bandwidth. This assumption is reasonable because the previous study [29] shows that the Internet backbone indeed has low utilization and the bottleneck almost happens at the parts near the end hosts. Based on the assumptions, we can develop the topology adaptation algorithm.

The topology adaptation algorithm is to set the upload rate for each upload connection to the fixed value, a rate r, for all peers and the seed S. Hence, if a peer i has maximum upload capacity of U_i, it establishes $k = \lceil \frac{U_i}{r} \rceil$ connections, where $r \leq U_i$, $\forall\ i \in \mathcal{N}$. Each peer establishes k concurrent upload connections among its working set, and intuitively a peer can upload the blocks it holds to other peers.

The basic idea behind this approach is that by serving k different peers with an uploading rate r simultaneously, the peer can fully utilize its upload capacity and thus maximize its contribution to the system throughput. For example, a peer with higher capacity might establish ten or more connections than a peer with lower capacity. However, a smaller value of r might slow down the distribution rate for blocks. The analysis of the upload rate r has been demonstrated in our previous work [30].

IV. SYSTEM ANALYSIS

In this section, we describe the $MDT(F)$ in the Bee can approach to the ideal dissemination time with a high probability. We also present the analysis for the Bee in terms of the scalability and efficiency. Here, we assume that all peers join the system at the same time and all the communications between peers are reliable. We also assume that peers do not leave the system either voluntarily or due to failures, and no transmission delay.

A. SCALABILITY

The design of Bee is very scalable. A peer only needs to maintain a random overlay mesh with a constant number of connections, regardless of the size of the system. This implies that each peer only connects to a few number of peers, so the loading in each peer should be very slight. The possible concern is the scalability of the register server in Bee. The register server in Bee serves as the same role as the tracker in BitTorrent or the rendezvous point in some application overlay multicast systems [31], [32]. The design of the register server can be distributed, the service loading is distributed evenly to many servers. Therefore, we believe that the register server should not be a critical problem to limit the scalability of Bee.

B. EFFICIENCY

We discuss why the $MDT(F)$ in the Bee can approach to the lower bound with a high probability. We do not provide a theoretical proof on the optimality of the Bee due to the inherent difficulty of any heuristic-driven distributed systems, such as BitTorrent. To the best of our knowledge, we are not aware of any theoretical results to prove that a distributed system can achieve the lower bound. In [33], Wu *et al.* show a centralized scheduling algorithm to minimize the dissemination time with all knowledge of system capacities. However, the centralized solution only works in a static network environment and it also does not consider the dynamic behaviors of peers in real systems.

Before we analyze Bee system, we assume that the goal of Bee is to disseminate a file F from a seed S to a number of receivers under the constraints of $\frac{size(F)}{U_s} \leq \overline{T}(F)$ and $\frac{size(F)}{\min\{D_i\}} \leq \overline{T}(F)$, where $\overline{T}(F)$ is defined in the Eq. 5. When the two constraints hold, neither the seed S nor the slowest peer does not become the bottleneck in the system. In addition, we also assume that the blocks are uniformly distributed among peers, which is caused by the local rarest first strategy.

Recall the analysis in Section II, we introduced two design principles for a data dissemination system to achieve the theoretical lower bound. The first one is that all peers should leave the system at the same time as much as possible, and each peer has to utilize its upload bandwidth as much as possible. For the first principle, the slowest peer strategy could force peers to progress at approximately the same download speed. For the other, all peers may fully contribute to the uploading capacity of whole system based on the topology adaptation algorithm, thus each peer can maintain its upload contribution to the system throughput continuously.

At the beginning of the system, only the seed S has the file, so it is impossible to fully utilize the upload bandwidth of each peer. We define the period of time for the system so that each peer has enough blocks to exchange as the start-up time of the system. Assume that the size of block is $256\tilde{}KB$, the start-up time is at about $\frac{256\tilde{}KB}{r} \times \log n$, where r is the upload rate and n is the number of peers in the system. Our previous work [18] shows how to find the optimal rate r in static networks. Moreover, a peer only sends out a block when it already received a request from a peer and it should not receive duplicated blocks.

Now, we provide an example to explain the behaviors in Bee. In homogeneous networks, the upload connections k of each peer is equivalent, assume $k = 5$ ($\frac{U_i}{r} = 5$). If the number of peers is n, the number of incoming connections per peer should be 5 ($5 = \frac{n \times k}{n}$) in average, due to the overlay mesh is constructed randomly. In heterogeneous networks, each peer

TABLE 1. The upload/download Bandwidth Distribution

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

in Bee could have the same number of incoming connections in average based the assumption. It should be easy to expound that Bee system could enforce most of peers at the same download progress approximately and make most of peers leaving the system at roughly the same time. Thus, with a high probability, the $MDT(F)$ of a Bee system can approximately approach to the ideal dissemination time both in homogeneous and heterogeneous networks.

V. PERFORMANCE EVALUATION

We made a simulation to compare the dissemination time in Bee with the lower bound of dissemination time and the required time in BitTorrent [6]. We consider two network scenarios, each representing a different degree of heterogeneity in their upload/download capacity. Table 1 presents the bandwidth distribution of each network condition. The heterogeneous network has two types of peers. A more heterogeneous condition with four types of peers is considered, and this setting is the realistic peer bandwidth distribution of Gnutella [34]. And the arrival pattern is flash crowd, i.e., all peers join at the initial stage and leave the system when they finish their downloading.

For the parameter r in Bee, we configure the uploading rate $r = 25$ in the all experiments based on our previous results [30]. All the experiments were run using an Intel Xeon E5560 CPU at 2.80 GHz with 512 GB of RAM. These source code of the simulations and the experimental results can be accessed publicly at https://github.com/cjwu/bee.

Unless otherwise specified, we use the following settings in our experiments. We used a file size of 200 MB with a block size 256 KB. The seed's uplink capacity is 6000 Kbps. The number of contact list is 40 in both Bee and BitTorrent, and the maximal number of concurrent upload connections per peer is 5 in BitTorrent settings. Then the number of initial seeds is only one in all of our experiments. Finally, the endgame model [35] of BitTorrent is not enabled because it only works for a small percentage of the download time.

A. HETEROGENEOUS ENVIRONMENT

We evaluate the performance of Bee and BitTorrent in a heterogeneous network that consists of two types of peers, one of which has a higher upload/download capacity (3000/1000 Kbps) than the other (1500/384 Kbps). In this scenario, the lower bound is 2333 seconds according to Eq. 6 in Section II.

Fig. 3 shows the comparisons of Bee to BitTorrent in heterogeneous environments. Here, we use a normalized MDT metric which is the MDT dividing the lower bound. Fig. 3(a) shows the normalized MDT metric for Bee and BitTorrent. In Fig. 3(a), we present the scalability of Bee by increasing the network size from 500 to 5000 in experiments. The results demonstrate that Bee is almost twice faster than BitTorrent in the MDT metric, and also show that both Bee and BitTorrent are scalable systems.

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 higher capacity peers leave BitTorrent, the total upload capacity of BitTorrent is decreased significantly, and the lower capacity peers are required to stay in the system longer to download the complete file. Thus, the MDT of BitTorrent is also prolonged, it fits our analysis in the Section II.

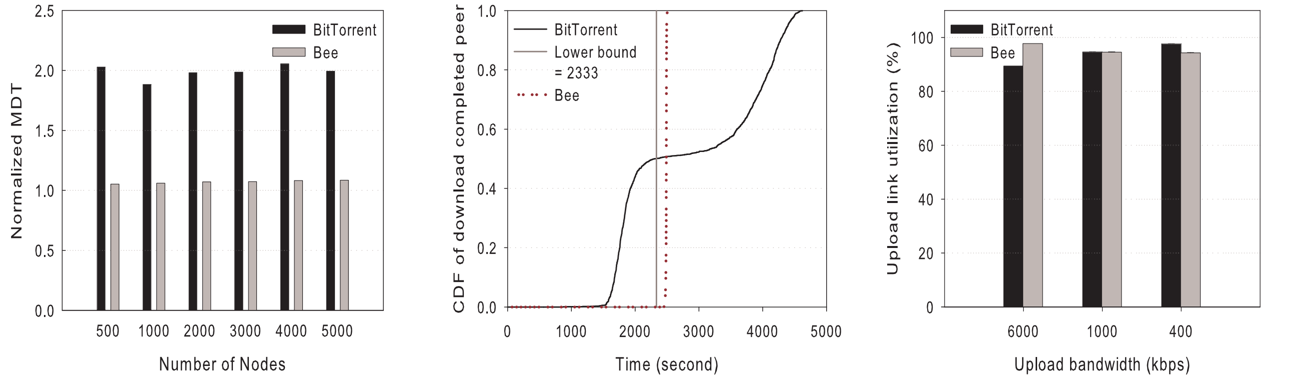
Compared to BitTorrent, the MDT of Bee approaches to the lower bound approximately due to the two design principles we analyzed in the section IV. More clearly, the normalized MDT of Bee is only 1.1. As we mentioned previously, any data dissemination system requires a start-up time to let peers have enough blocks to exchange and to utilize their uplink capacities. The results examine that the start-up time of an efficient system could be short.

In addition, we show the average uploading link utilization of peers, including the seed (6000 Kbps) and two type of peers (1000 Kbps and 400 Kbps) in Fig. 3(c). The result shows that the uplink utilization of each peer is over 90% (96% in Bee) in average, which means that the overall upload utilizations of the two systems are close to fully utilized. However, in BitTorrent, a peer with higher upload capacity should exchange blocks with another one with similar upload capacity, because the Tit-For-Tat (TFT) peer selection strategy [5] is likely to reward for the one with similar upload capacity. As a result, the overall uplink utilization of BitTorrent is efficient, but the design philosophy of BitTorrent is exclusively due to egoistic motivation, so the lower capacity peers need more time to download the complete file.

B. MORE HETEROGENEOUS ENVIRONMENT

In this section, we repeat the above experiments in a more heterogeneous network with four types of peer capacities, the detailed bandwidth distribution is presented in Table I. This results examine the behaviors of Bee and BitTorrent in a complex network environment and in a real P2P network condition. In this simulation setting, the lower bound is 2089 seconds, it can be derived from the Eq. 6 ($\frac{size(F)}{\min\{D_i\}} = \frac{1638400}{784}$). Thus, the bottleneck of the system is at the download link of the slowest peer.

First, we study the impacts of various network sizes by scaling from 500 to 5000 peers. Fig. 4(a) shows the normalized MDT metric of Bee and BitTorrent. This result also shows that Bee is at least two times faster than the BitTorrent in MDT metric. As a result, the bottleneck of the system (the

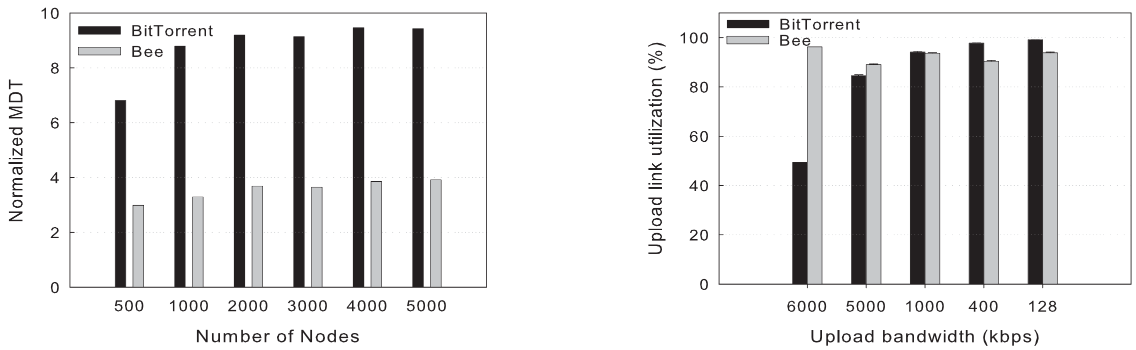


(a) Comparison of Bee to BitTorrent for *MDT*.

(b) Cumulative distribution of download peers on Bee and BitTorrent with 2000 nodes.

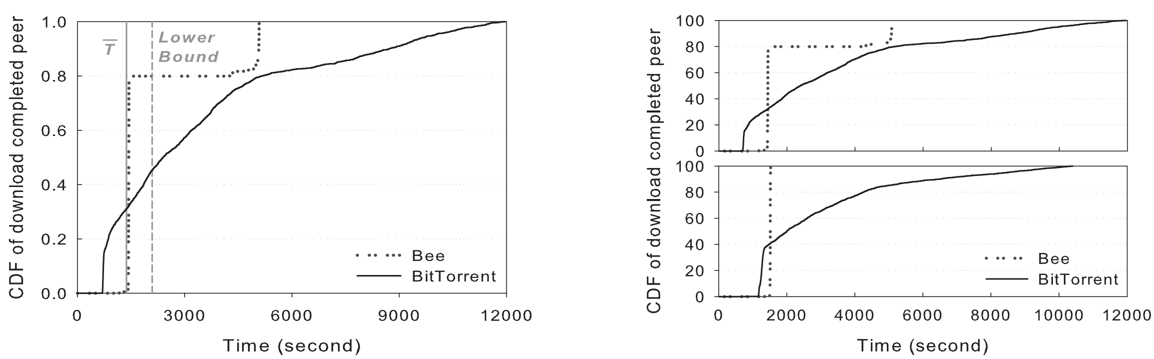
(c) The average uploading link utilization of peers, seed (6000Kbps), higher capacity peers (1000Kbps) and lower capacity peers (400Kbps).

FIG. 3. **The comparisons of Bee to BitTorrent in the heterogeneous environment.**



(a) Comparison of Bee to BitTorrent for *MDT*

(b) The average uploading link utilization of peers, seed (6000Kbps), others indicate peer capacity



(c) Cumulative distribution of download peers on Bee and BitTorrent with 2000 peers.

(d) Cumulative distribution of peers on Bee that each peer can download a complete file before the lower bound.

FIG. 4. **The comparisons of Bee to BitTorrent in the more heterogeneous environment.**

slowest peer) makes a significant impact on the *MDT* metric. Without a doubt, the result shows that both Bee and BitTorrent are scalable systems again, even in a more heterogeneous network.

Next, we examine the uplink utilization of peers in Bee and BitTorrent in the more heterogeneous network. Fig. 4(b) shows the uplink utilization of peers in Bee and BitTorrent. The results indicate that the uplink utilization of the seed in Bee is over 90%, but only 50% in BitTorrent. From our viewpoints, in BitTorrent, when the variance of upload capacity increases, more peers with higher capacity will leave the system early. So that the seed's uplink utilization may be limited, because the download capacity of the lower capacity peers is too small to fully utilize the uplink capacity of the seed. That might be the main reason that the uplink utilization of the seed in BitTorrent is only 50% in average, especially the upload connections of the seed is fixed to 5 in original BitTorrent setting. As a result, the uplink utilization of the seed decreases when the variance of download capacity increases in BitTorrent. However, the uplink utilization of seed in Bee can be fully utilized regardless of the heterogeneity degree of uplink capacity, it benefits from the topology adaptation algorithm of Bee. In Fig. 4(b), BitTorrent performs better than Bee when the upload bandwidth is less than 1000 kbps. The main reason is that the lower capacity nodes ($<$ 1000 kbps) in Bee need more time to find the nodes to contribute their upload bandwidth to the system, especially in this simulation settings.

We now investigate the MDT metric of Bee and BitTorrent in the complex network environment. In Fig. 4(c), we show the cumulative distribution of the number of complete peers in the complex network with 2000 peers. Fig. 4(c) illustrates that 80% peers leave Bee system at the $\overline{T}$ time (Recall that the Eq. 6) and the remained 20% peers prolong the MDT of Bee system. More clearly, these 80% peers are higher capacity peer (download capacity), and the dissemination time of remained peers (poor download capacity) is limited by their download capacities. Note that in this simulation, the bottleneck of the system is at the download link of the slowest peer. Again, the result shows that when the higher capacity peers leave system early, the upload capacity of overall system decreases dramatically, and that results in a longer maximum dissemination time in both Bee and BitTorrent. However, BitTorrent needs more than eight times to finish downloading compared to the lower bound.

Next, we configure the download capacities of all peers to make sure that the lower bound is not at the download capacity of peers ($\frac{size(F)}{\min\{D_i\}}$) and repeat the simulation again to examine the MDT metric of Bee and BitTorrent. We only increase the download capacity of the slowest peers from 784 Kbps to 1200 Kbps in this network. So that the lower bound in this network is $\overline{T}$ after the configurations are applied, here the ($\frac{size(F)}{\min\{D_i\}} = 1365.3$) is smaller than ($\overline{T} = 1374.9$).

Fig. 4(d) shows the cumulative distribution of the number of complete peers with the configurations. In Fig. 4(d), the top figure is the CDF with 784 Kbps (minimum download capacity) and the bottom one is the CDF with 1200 Kbps. As shown as the results, we can observe that the MDT in Bee can approximately approach the lower bound $\overline{T}$. The result implies that the performance of Bee is efficient and BitTorrent might be the network heterogeneity independent. The result also shows that the dissemination time in Bee can approach the lower bound approximately when the bottleneck is not at the download capacity of all peers. However, increasing the download capacity of all peers does not improve the MDT metric of BitTorrent. The result also shows that it still is a long-tail curve in the results of BitTorrent regardless of the bottleneck is at the download capacities or not.

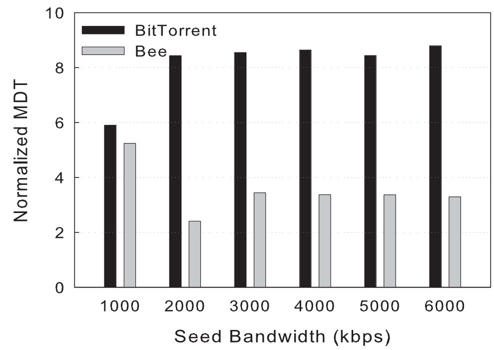


FIG. 5. The comparison of Bee to BitTorrent in various uplink capacities of the seed.

C. THE IMPACT ON SEED CAPACITY

We consider the effects of various seed capacities on the performance of Bee and BitTorrent. The number of peers at the initial stage is set to 2000. In Fig. 5, Bee obviously outperforms BitTorrent in the performance index MDT. However, when the uplink capacity of the seed drops to 1000 Kbps, the bottleneck of this system is at the uplink capacity of the seed ($\frac{size(F)}{U_s}$), so the MDT of Bee and BitTorrent both result in poor performance. This result implies that the efficiency of Bee is limited by the seed's capacity, but BitTorrent has little effect on the seed's capacity. All peers in BitTorrent have to stay in the system until all blocks have been spread to the system, it improves the MDT performance of BitTorrent. These results also meet our system analysis in Section IV, a poor capacity seed may make a dissemination system require more start-up time to let peers have enough blocks to stabilize exchanging process and inhibits development of spreading blocks.

D. THE IMPACT ON ARRIVAL PATTERN

We evaluate the effects of various arrival rates for Bee and BitTorrent in the more heterogeneous network conditions (with four types of peer capacities). Fig. 6 shows the normalized MDT performance comparisons of Bee and BitTorrent in various arrival rates. In Fig. 6, we can observe that when the arrival rate is low (0.1), a few peers join the system and contribute upload capacity, so that the overall upload capacity of the system is poor. And another reason is that more peers may need to wait at a longer **start-up time** to receive blocks and to exchange blocks. So the peers in Bee or BitTorrent would require more time to complete the download file. However, as peer arrival rate is increased, the upload capacity of the system also is increased promptly, that result corresponds to

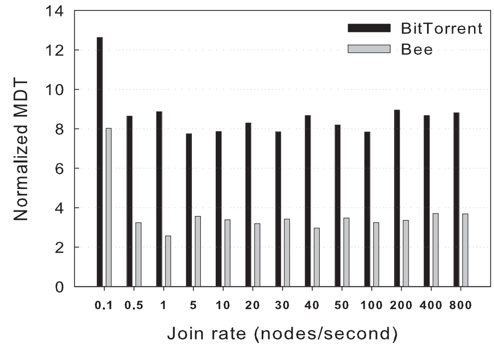


FIG. 6. The comparison of Bee to BitTorrent in various arrival rate.

the analysis in Section IV. Based on the results, Bee outperforms better performance than BitTorrent in MDT metrics regardless of the arrival rate.

E. THE IMPACT ON FLASH CROWD TRAFFIC

We now evaluate Bee and BitTorrent in a realistic flash crowd traffic. In this experiment, each peer joins the system according to the tracker log of a Redhat 9 distribution torrent [36]. The capacity of each peer is randomly assigned with one of four types of peer capacities, and the upload bandwidth of the seed is set to 6000 Kbps. Note that the lower bound in this case is $\frac{size(F)}{\min\{D_i\}} = 2089$ seconds.

All the results are shown in Fig. 7. Fig. 7(a) illustrates the distribution of peers arrival time, the tracker log consists of over 12,000 peers arrival time, almost 80% of which arrived before 2000 minute. When a new version of Redhat IOS is released, the flash crowd phenomenon occurs at the file release time, numerous users suddenly request to download the file. The flash crowd traffic model captures the most prominent flash crowd characteristics observed in these traces.

First, we show the download completion time of each peer for Bee and BitTorrent in Fig. 7(b). The result shows that 83% peers in Bee finish their download before 2000 seconds. On the other hand, only 50% BitTorrent peers can complete their download at 2000 seconds. Note that the lower bound is at the poor download capacity peers (the lower bound is inherently limited $\frac{size(F)}{\min\{D_i\}}$). So these poor peers need more download time to complete the file, and the higher capacity peers (the lower bound is $\overline{T}$) can receive the complete file and leave the system without waiting these slow download peers. So at the 10,000 time point, most of peers remained in the system (17%) are poor download capacity peers, this is the same behavior we found in Fig. 4(c). The result also shows that the distribution of the MDT of BitTorrent still is a long-tail curve by the same process in Fig. 4(c). Moreover, compared to BitTorrent, Bee only needs 1/3 time to finish the file dissemination in the more heterogeneous network.

Here, we demonstrate the uplink utilization of all peers in Bee and BitTorrent in Fig. 7(c) and Fig. 7(d), respectively. We can see that the uplink utilization of each peer in Bee is fully utilized (94% in most of peers). And we see that BitTorrent results most of the time in a very poor uplink utilization. One reason for this should be the TFT peer selection strategy of BitTorrent, it might pair a higher uplink capacity peer with a lower download capacity peer, and TFT strategy keeps to search the peer with better upload contributions. During the peer searching process, the uplink capacity of the peer is at low utilization. Thus the higher uplink capacity peers can not contribute their full uplink bandwidth to the system continuously. The main reason is that BitTorrent is inherently limited by its design principles, which encourages fairness [37] in peers, the TFT strategy makes all peers achieve fairness in BitTorrent, considers to give and take equitably.

In summary, based on the simulation results, we show that Bee has the ability to roughly approximate the lower bound of a data dissemination system even in complex network scenarios if and only if the lower bound of the system is not at uplink capacity of the seed ($\frac{size(F)}{U_s}$) and the download capacity of the peers ($\frac{size(F)}{\min\{D_i\}}$). Bee is suitable as a building block in a time-critical data dissemination applications, which can be the virtual machine (VM) deployment in cloud computing platforms [38] or distributing the urgent content in network security events [39].

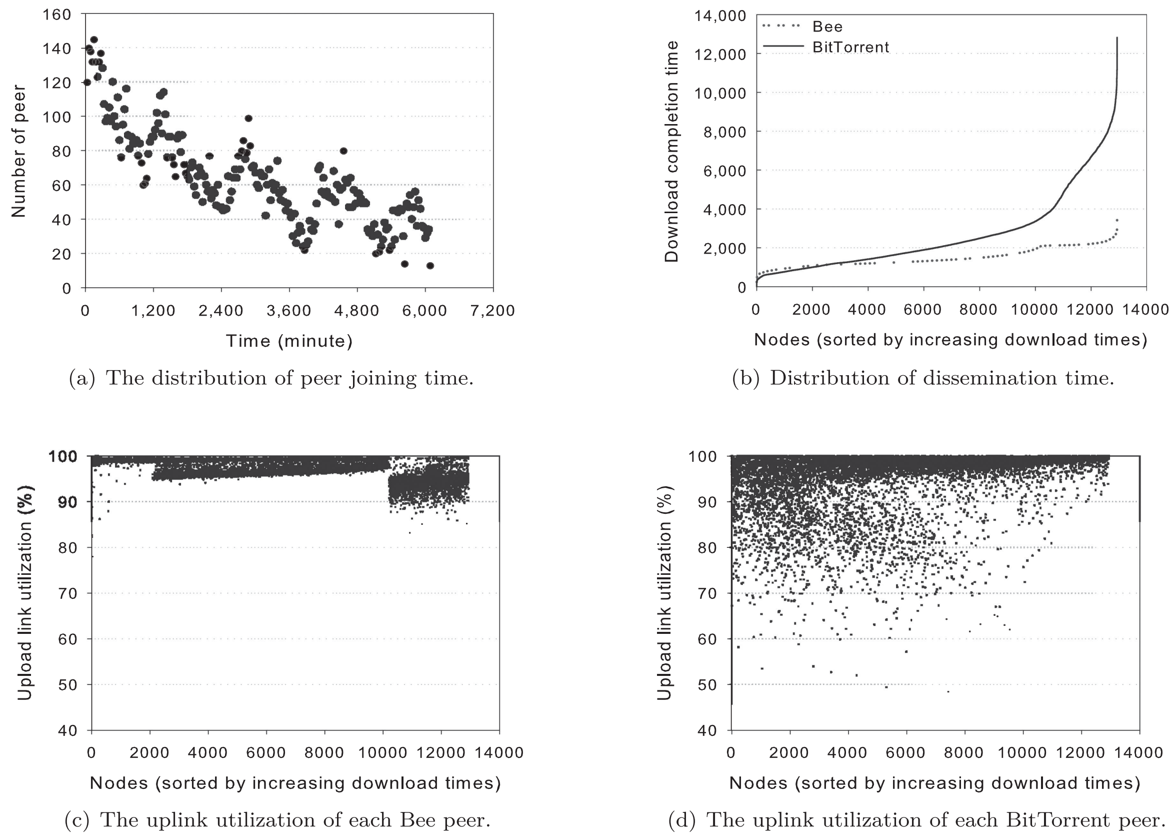
VI. RELATED WORK

In recent years, there are tremendous interests in building content delivery systems [40] to distribute content, which aims to deliver large-sized data to a large group of nodes spread across a wide-area network. However, how to design an efficient data dissemination system to achieve the lower bound of data dissemination time has not been discussed in the previous literatures, especially under flash crowd traffic. In this section, we describe the recent research works about flash crowd traffic and present the related work on the peer-assisted content delivery systems. In addition, there are some advanced coding techniques [41], [42] for content delivery, the detailed survey can be reached in the article [43].

A. FLASH CROWD TRAFFIC

A flash crowd is a large traffic surge to a particular system. It is not an usual event but causes poor performance at the system and results in a significant number of unsatisfied users. When a flash crowd occurs, the sudden arrival of numerous peers may starve the capacity of a system, and degrade the quality of service. Thus, it is important to understand the challenges for a data dissemination system, and how flash crowds affect the efficiency of data dissemination.

There is a large body of work on the modeling of P2P data dissemination systems [44] but a few work focusing on flash crowd [1], [45], [46]. In reality, flash crowds may result the worst case performance of P2P data dissemination systems [1]. Zhang *et al.* [1] propose a model for analyzing BitTorrent flashcrowds by studying millions of swarms from BitTorrent trackers. They show BitTorrent flashcrowds occur in very small fractions, but affect over million users. Carbunaru *et al.* [45] formulate an analytical model for flash



(a) The distribution of peer joining time.

(b) Distribution of dissemination time.

(c) The uplink utilization of each Bee peer.

(d) The uplink utilization of each BitTorrent peer.

FIG. 7. Performance of Bee and BitTorrent with arrival rate from Redhat 9 tracker log.

crowds in homogeneous and heterogeneous capacity networks and focused on performance scalability of content distribution and server provisioning during flash crowds. Chen *et al.* [46] provide a fluid model study on the performance of P2P live streaming systems under flash crowds, and denote that the worst-case peer startup latency and system recovery time increase logarithmically with the flash crowd size. A key difference is that we analyze the impact of multiple classes of peers on heterogeneous networks to achieve the lower bound of the maximum dissemination time during flash crowd.

B. PEER-ASSISTED CONTENT DELIVERY SYSTEMS

The peer-assisted content delivery systems have received a lot of attention from Internet users and networking researchers [47]–[50]. Interested reader can refer to Nasreen *et al.*, [51] who present a detailed survey on this topic. The main concept of the peer-assisted content delivery is inspired from the parallel-downloading mechanism [52]. BitTorrent [6] is a popular content distribution system which is successful for its efficiency in delivering a large file. There are two mechanisms used in BitTorrent, namely, the TFT peer selection policy and the local rarest first piece selection strategy. Slurpie [7] focuses on reducing loading on servers and peer download times. Slurpie uses an adaptive downloading mechanism to improve peer's performance according to its capacity, and adopts a random back-off algorithm to control loading on the server. Crew [53] is a gossip-based system for data dissemination and it performs better dissemination performance than BitTorrent in experiments, but how close it approaches to the lower bound is still unknown. Kumar *et al.* [54] demonstrate a set of expressions for the minimum distribution time of a general heterogeneous peer-assisted file distribution system. Ezovski *et al.* [55] provide an analytical result of minimizing average finish time by using the water-filling technique, in an upload-constrained P2P network. The research results focused on analyzing and minimizing the download time of a single peer is presented in [56] and minimizing the average download time of a system is presented in [57]. Zheng *et al.* [58] formulate an optimization problem of content distribution as an optimal set of distribution trees for determining the rate of distribution on each tree under bandwidth limitation networks.

VII. CONCLUSION

We define the data dissemination problem and examine an analysis of the lower bound of the maximum dissemination time for this problem under flash crowd traffic. We present the design principles of Bee to capture the two following notions: 1) all peers stay the system to contribute their upload capacities (the slowest peer selection strategy), and 2) all peers fully utilize their upload capacities (the topology adaptation algorithm). Bee does not require any scheduling knowledge, each peer makes its own decision to download blocks based on the local knowledge. We have conducted extensive simulations to evaluate the MTD performance of Bee, and the results offer evidence that the maximum dissemination time in Bee can roughly approximate the lower bound when there are no bottleneck at the seed or at the download capacity of the slowest peer, even under flash crowd traffic. In the current Internet or the cloud computing platforms, Bee can play a major role for addressing the time-critical data dissemination applications in future development. It would be interesting to examine the performance of Bee in the current live streaming applications [59], especially in the mobile internet [60]. We will implement and deploy the Bee system in cloud computing platforms for investigating the performance on the VM deployment in our future work.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Kuan-Ta Chen for comments on an earlier draft of this paper, and this paper is dedicated to the memory of our dear Dr. Kuan-Ta Chen and the anonymous reviewers for their valuable comments and suggestions to improve the manuscript.

REFERENCES

[1] B. Zhang, A. Iosup, J. Pouwelse, and D. Epema, “Identifying, analyzing, and modeling flashcrowds in bittorrent,” in *Proc. IEEE Int. Conf. Peer-to-Peer Comput.*, 2011, pp. 240–249.

[2] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, “Data-driven computing and caching in 5G networks: Architecture and delay analysis,” *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 70–75, Feb. 2018.

[3] X. Li, X. Wang, C. Zhu, W. Cai, and V. C. M. Leung, “Caching-as-a-service: Virtual caching framework in the cloud-based mobile networks,” in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2015, pp. 372–377.

[4] A. Yahyavi and B. Kemme, “Peer-to-peer architectures for massively multiplayer online games: A survey,” *ACM Comput. Surv.*, vol. 46, no. 1, 1–51, 2013.

[5] Y. Nishi, M. Sasabe, and S. Kasahara, “Optimality analysis of locality-aware tit-for-tat-based P2P file distribution,” *Peer-to-Peer Netw. Appl.*, vol. 13, no. 5, pp. 1688–1703, Sep. 2020.

[6] B. Cohen, “Incentives build robustness in bittorrent,” in *Proc. Workshop Econ. Peer-to-Peer Syst.*, 2003, pp. 68–72.

[7] R. Sherwood, R. Braud, and B. Bhattacharjee, “Slurpie: A cooperative bulk data transfer protocol,” in *Proc. IEEE 23rd Annu. Joint Conf. IEEE Comp. Commun. Societies*, vol. 2, 2004, pp. 941–951.

[8] D. Kostić *et al.*, “High-bandwidth data dissemination for large-scale distributed systems,” *ACM Trans. Comput. Syst.*, vol. 26, no. 1, pp. 1–61, 2008.

[9] K. Kim, S. Mehrotra, and N. Venkatasubramanian, “Efficient and reliable application layer multicast for flash dissemination,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2571–2582, Oct. 2014.

[10] M. Deshpande, K. Kim, B. Hore, S. Mehrotra, and N. Venkatasubramanian, “ReCREW: A reliable flash-dissemination system,” *IEEE Trans. Comput.*, vol. 62, no. 7, pp. 1432–1446, Jul. 2013.

[11] M. Zghaibeh, “O-torrent: A fair, robust, and free riding resistant P2P content distribution mechanism,” *Peer-to-Peer Netw. Appl.*, vol. 11, no. 3, pp. 579–591, May 2018.

[12] C.-J. Wu, C.-Y. Li, and J.-M. Ho, “Improving the download time of bittorrent-like systems,” in *Proc. IEEE Int. Conf. Commun.*, 2007, pp. 1125–1129.

[13] B. Barekatain *et al.*, “Matin: A random network coding based framework for high quality peer-to-peer live video streaming,” *PLoS One*, vol. 8, no. 8, pp. 1–17, 2013, Art. no. ee69844.

[14] X. Yang and G. de Veciana, “Service capacity of peer to peer networks,” in *Proc. IEEE Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 4, 2004, pp. 2242–2252.

[15] N. Khan, M. Moharrami, and V. Subramanian, “Stable and efficient piece-selection in multiple swarm bittorrent-like peer-to-peer networks,” in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1153–1162.

[16] B. Fan, J. C. S. Lui, and D. Chiu, “The design trade-offs of bittorrent-like file sharing protocols,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 365–376, Apr. 2009.

[17] 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.

[18] C.-J. Wu, C.-F. Ku, J.-M. Ho, and M.-S. Chen, “A novel pipeline approach for efficient Big Data broadcasting,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 17–28, Jan. 2016.

[19] Z. Xu *et al.*, “Energy-aware collaborative service caching in a 5G-enabled MEC with uncertain payoffs,” *IEEE Trans. Commun.*, vol. 70, no. 2, pp. 1058–1071, 2022, doi: 10.1109/TCOMM.2021.3125034.

[20] Q. Cheng, H. Shan, W. Zhuang, L. Yu, Z. Zhang, and T. Q. S. Quek, “Design and analysis of MEC- and proactive caching-based 360 mobile VR video streaming,” *IEEE Trans. Multimedia*, pp. 1–1, Mar. 19, 2021, doi: 10.1109/TMM.2021.3067205.

[21] A. M. Farley, “Broadcast time in communication networks,” *SIAM J. Appl. Math.*, vol. 39, no. 2, pp. 385–390, 1980.

[22] S. Khuller and Y.-A. Kim, “Broadcasting in heterogeneous networks,” *Algorithmica*, vol. 48, no. 1, pp. 1–21, May 2007.

[23] M. Deshpande, N. Venkatasubramanian, and S. Mehrotra, “Heuristics for flash-dissemination in heterogenous networks,” in *Proc. 13th Int. Conf. High Perform. Comput.*, 2006, pp. 607–618.

[24] K.-S. Goetzmann, T. M. Harks Klimm, and K. Miller, “Optimal file distribution in peer-to-peer networks,” in *Algorithms and Computation*, T. Asano, S.-i. Nakano, Y. Okamoto, and O. Watanabe, Eds. Berlin, Heidelberg:Springer Berlin Heidelberg, 2011, pp. 210–219.

[25] W.-C. Liao, F. Papadopoulos, K. Psounis, and C. Psomas, “Modeling bittorrent-like systems with many classes of users,” *ACM Trans. Model. Comput. Simul.*, vol. 23, no. 2, pp. 1–25, May 2013.

[26] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, “Analyzing and improving a bittorrent networks performance mechanisms,” in *Proc. IEEE INFOCOM 25th IEEE Int. Conf. Comput Commun.*, 2006, pp. 1–12.

[27] A. Botta, G. E. Mocerino, S. Cilio, and G. Ventre, “A machine learning approach for dynamic selection of available bandwidth measurement tools,” in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[28] J. Strauss, D. Katabi, and F. Kaashoek, “A measurement study of available bandwidth estimation tools,” in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas.*, 2003, pp. 39–44.

[29] A. Akella, S. Seshan, and A. Shaikh, “An empirical evaluation of wide-area internet bottlenecks,” in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas.*, 2003, pp. 101–114.

[30] C. Wu, C. Li, K. Yang, J. Ho, and M. Chen, “Time-critical data dissemination in cooperative peer-to-peer systems,” in *Proc. IEEE Glob. Telecommun. Conf.*, 2009, pp. 1–6.

[31] Y.-h. Chu, S. G. Rao, S. Seshan, and H. Zhang, “A case for end system multicast,” *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[32] C.-J. Wu, D.-K. Liu, and R.-H. Hwang, “A location-aware peer-to-peer overlay network,” *Int. J. Commun. Syst.*, vol. 20, no. 1, pp. 83–102, 2007.

[33] G. Wu and T. C. Chiueh, “How efficient is BitTorrent?,” in *Int. Soc. Opt. Photonics. SPIE*, S. Chandra and C. Griwodz, Eds., in Multimedia Computing and Networking 2006, vol. 6071, 2006, pp. 266–278.

[34] S. Saroiu, P. K. Gummadi, and S. D. Gribble, “Measurement study of peer-to-peer file sharing systems,” in *Multimedia Computing Networking* 2002, M. G. Kienzle and P. J. Shenoy, Eds., vol. 4673, International Society for Optics and Photonics. SPIE, 2001, pp. 156–170.

[35] A. Legout, G. Urvoy-Keller, and P. Michiardi, “Rarest first and choke algorithms are enough,” in *Proc. 6th ACM SIGCOMM Conf. Internet Meas.*. New York, NY, USA: Association for Computing Machinery, 2006, pp. 203–216.

[36] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, A. Al Hamra, and L. Garcés-Erice, “Dissecting bittorrent: Five months in a torrent’s lifetime,” in *Passive and Active Network Measurement*, C. Barakat and I. Pratt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–11.

[37] A. Sherman, J. Nieh, and C. Stein, “Fairtorrent: A deficit-based distributed algorithm to ensure fairness in peer-to-peer systems,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1361–1374, Oct. 2012.

[38] M. Schmidt, N. Fallenbeck, M. Smith, and B. Freisleben, "Efficient distribution of virtual machines for cloud computing," in *Proc. 18th Euromicro Conf. Parallel, Distrib. Network-based Process.*, 2010, pp. 567–574.
[39] L. Bilge and T. Dumitraş, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conf. Comput. Commun. Secur.*, New York, NY, USA: Association for Computing Machinery, 2012, p. 833–844.
[40] B. Zolfaghari *et al.*, "Content delivery networks: State of the art, trends, and future roadmap," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–34, Apr. 2020.
[41] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 4, 2005, pp. 2235–2245.
[42] J. Su, Q. Deng, and D. Long, "Pclnc: A low-cost intra-generation network coding strategy for P2P content distribution," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 1, pp. 177–188, Jan. 2019.
[43] B. Li and D. Niu, "Random network coding in peer-to-peer networks: From theory to practice," *Proc. IEEE Proc. IRE*, vol. 99, no. 3, pp. 513–523, Mar. 2011.
[44] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proc. Conf. Appl., Technol., Architectures, Protoc. Comput. Commun.*. New York, NY, USA: Association for Computing Machinery, 2004, pp. 367–378.
[45] C. Carbunaru, Y. M. Teo, B. Leong, and T. Ho, "Modeling flash crowd performance in peer-to-peer file distribution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2617–2626, Oct. 2014.
[46] Y. Chen, B. Zhang, C. Chen, and D. M. Chiu, "Performance modeling and evaluation of peer-to-peer live streaming systems under flash crowds," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1106–1120, Aug. 2014.
[47] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Comput. Netw.*, vol. 116, pp. 79–95, 2017.
[48] P. Michiardi, D. Carra, F. Albanese, and A. Bestavros, "Peer-assisted content distribution on a budget," *Comput. Netw.*, vol. 56, no. 7, pp. 2038–2048, 2012.
[49] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?," in *Proc. 5th ACM SIGCOMM Conf. Internet Meas.*, USA: USENIX Association, 2005, pp. 6–6.
[50] J. Lin, Z. Li, G. Xie, Y. Sun, K. Salamatian, and W. Wang, "Mobile video popularity distributions and the potential of peer-assisted video delivery," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 120–126, Nov. 2013.
[51] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Comput. Netw.*, vol. 116, pp. 79–95, 2017.
[52] P. Rodriguez and E. W. Biersack, "Dynamic parallel access to replicated content in the internet," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, p. 455–465, Aug. 2002.
[53] M. Deshpande, B. Xing, I. Lazardis, B. Hore, N. Venkatasubramanian, and S. Mehrotra, "Crew: A gossip-based flash-dissemination system," in *Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst.*, 2006, pp. 45–45.
[54] R. Kumar and K. W. Ross, "Peer-assisted file distribution: The minimum distribution time," in *Proc. 1st IEEE Workshop Hot Topics Web Syst. Technol.*, 2006, pp. 1–11.
[55] G. M. Ezovski, A. Tang, and L. L. H. Andrew, "Minimizing average finish time in P2P networks," in *Proc. IEEE INFOCOM*, 2009, pp. 594–602.
[56] Y.-M. Chiu and D. Y. Eun, "Minimizing file download time in stochastic peer-to-peer networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 253–266, Apr. 2008.
[57] M. Sasabe, "Analysis of minimum distribution time of tit-for-tat-based P2P file distribution: Linear programming based approach," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 4, pp. 2127–2138, 2021.
[58] X. Zheng, C. Cho, and Y. Xia, "Content distribution by multiple multicast trees and intersession cooperation: Optimal algorithms and approximations," *Comput. Netw.*, vol. 83, pp. 100–117, 2015.
[59] X. Wei, P. Ding, L. Zhou, and Y. Qian, "QoE oriented chunk scheduling in P2P-VoD streaming system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8012–8025, Aug. 2019.
[60] M. Qin, L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Computing and relaying: Utilizing mobile edge computing for P2P communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1582–1594, Feb. 2020.

CHI-JEN WU (Member, IEEE) received the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in July 2012. From 2012 to 2013, he was a Distinguished Postdoctoral Scholar with the Institute of Information Science, Academia Sinica, Taipei, Taiwan. Since February 2021, he has been an Assistant Professor with the Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan City, Taiwan. His research interests include content distribution, mobile cloud computing, and artificial intelligence with a specific focus on computational advertising, marketing automation, and financial computing.

JAN-MING HO (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from Northwestern University, Evanston, IL, USA, in 1989. In 1989, he joined the Institute of Information Science, Academia Sinica, Taipei, Taiwan, as an Associate Research Fellow and was promoted to a Research Fellow in 1994. His research interests include the integration of theory and applications, including information retrieval and extraction, knowledge management, combinatorial optimization, multimedia network protocols and their applications, web services, bioinformatics, and digital library and archive technologies. Dr. Ho also published results in VLSI/CAD physical design.