A Novel Pipeline Approach for Efficient Big Data Broadcasting

Chi-Jen Wu, Chin-Fu Ku, Jan-Ming Ho, *IEEE Senior Member,* Ming-Syan Chen, *IEEE Fellow*

Abstract—Big-Data Computing is a new critical challenge for the ICT industry. Engineers and researchers are dealing with data sets of petabyte scale in the cloud computing paradigm. Thus the demand for building a service stack to distribute, manage and process massive data sets has risen drastically. In this paper, we investigate the Big Data Broadcasting problem for a single source node to broadcast a big chunk of data to a set of nodes with the objective of minimizing the maximum completion time. These nodes may locate in the same datacenter or across geo-distributed datacenters. This problem is one of the fundamental problems in distributed computing and is known to be NP-hard in heterogeneous environments. We model the Big-data broadcasting problem into a LockStep Broadcast Tree (LSBT) problem. The main idea of the LSBT model is to define a basic unit of upload bandwidth, r, such that a node with capacity c broadcasts data to a set of $\lfloor c/r \rfloor$ children at the rate r. Note that r is a parameter to be optimized as part of the LSBT problem. We further divide the broadcast data into m chunks. These data chunks can then be broadcast down the LSBT in a pipeline manner. In a homogeneous network environment in which each node has the same upload capacity c, we show that the optimal uplink rate r^* of LSBT is either $c/2$ or $c/3$, whichever gives the smaller maximum completion time. For heterogeneous environments, we present an $O(nlog^2n)$ algorithm to select an optimal uplink rate r^* and to construct an optimal LSBT. Numerical results show that our approach performs well with less maximum completion time and lower computational complexity than other efficient solutions in literature.

Keywords—Big data computing, data delivery algorithm, cloud computing, distributed computing, big data management.

1 Introduction

Big-Data Computing is a new critical challenge that has sparked major research efforts to reshape ICT industry and scientific computing in the past few years [1]. The rapid advances in ICT technologies, such as computation, communication and storage have resulted in enormous data sets in business, science and society being generated and analyzed to explore the value of those data. Currently, both ICT industry engineers and scientific researchers are dealing with petabytes of data sets in the cloud computing paradigm [2]. For instance, in industry, Google, Yahoo!, and Amazon collect huge amount of data every day for providing information services freely to people in useful ways. In science, the Large Hadron Collider (LHC) can generate about fifteen petabytes of data annually, and thousands of scientists around the world need to access and analyze those big data sets [3]. Thus the demand for building a distributed service stack to efficiently distribute, manage and to process massive data sets has risen drastically.

- *Chi-Jen Wu is with the Department of Electrical Engineering, National Taiwan University, Taiwan and also with the Institute of Information Science, Academia Sinica, Taiwan.*
  E-mail: cjwu@arbor.ee.ntu.edu.tw
- *Dr. Chin-Fu Ku is with the Institute of Information Science, Academia Sinica, Taiwan. E-mail: chinfu@iis.sinica.edu.tw*
- *Dr. Jan-Ming Ho is with the Institute of Information Science, Academia Sinica, Taiwan. and also with the Research Center of Information Technology Innovation, Academia Sinica, Taiwan*
  E-mail: hoho@iis.sinica.edu.tw
- *Dr. Ming-Syan Chen is with the Research Center of Information Technology Innovation, Academia Sinica, Taiwan and also with the Department of Electrical Engineering, National Taiwan University, Taiwan.*
  E-mail: mschen@cc.ee.ntu.edu.tw

In the past decade, several efficient techniques are proposed to manipulate huge amount of data, ranging from terabytes to petabytes, on as many as tens of thousands of machines. For example, Google presented a distributed computing framework, namely MapReduce [4], to process large-scale data effectively, and also proposed Bigtable [5] for storing structured data on thousands of machines. These techniques allows users to realize data-parallelism [6]. There are many significant issues in developing MapReduce applications, such as, designing the effective strategy for data decomposition, load balancing, and exchanging data among a large set of nodes [7]. In particular, for big-data computing, data transmission overhead is a significant factor of the job completion time. For instance, it is shown that the total amount of data transmission time occupies approximately one-third of the jobs' running time in the Hadoop tracing logs of Facebook [8].

In this paper, we focus on the big data broadcasting operation that is one of the most essential communication mechanisms in distributed systems. There are a lot of application domains that widely apply broadcasting operations, such as scientific data distributions [9], database transaction logs backups, the latest security patches, multimedia streaming applications, and data replica or virtual appliance deployment [10] among distributed data centers. Since the size of data becomes so enormous, the impact of broadcasting operation also becomes increasingly significant.

We consider the big data broadcasting problem in a heterogeneous network where nodes may have different uploading capacities. The big data broadcasting problem is about how the nodes may obtain a given big data cooperatively in a minimum amount of total transmission time. We assume that there are n nodes in a heterogeneous network system, denoted by n_1, n_2, n_3,$\ldots$,n_n. And the node n_1 is the broadcasting source that has the data item divided into m chunks of equal size, it disseminates the data item to all the other nodes. The upload capacities of those nodes are denoted by c_1, c_2, c_3,$\ldots$,c_n, measured in kilobyte per second (KBps). In addition, we assume that the downloading capacity of each node is larger than or equal to its uploading capacity.

Specifically, we focus on investigating the following questions: *What is the relation between a single overlay tree with a fixed uplink rate and the broadcast operation itself, and how to construct a single overlay tree that minimizes the maximum completion time in heterogeneous networks*? We introduce the novel LockStep Broadcast Tree (LSBT) to model the Big Data broadcast problem [11], [12]. LSBT is a broadcast tree where data chunks can be sent in a pipelined fashion with a good throughput. The main idea is to define a basic unit of upload bandwidth, r, such that the upload link of each node is divide into several connections each being allocated with the bandwidth r in broadcasting. In so doing, the number of upload connections is proportional to the capacity of a node. Furthermore, we also divide the broadcast data into m chunks. These data chunks are then broadcast down the tree by the nodes in a ***pipeline*** manner. We show that based on the LSBT model, the maximum number of rounds required to complete the broadcast of entire data chunks is O$(m + logn)$ steps, where n is the number of nodes. In a homogeneous network environment in which each node has the same uploading capacity c, we show that the optimal uplink rate r^* of LSBT is either $c/2$ or $c/3$. For heterogeneous networks, we present an O$(nlog^2n)$ algorithm to select an optimal uplink rate r^* and to construct an optimal LSBT. Numerical results show that the maximum completion time of our LSBT approximates to the optimum of the big data broadcast problem.

The main contributions of this paper are as follows.

1) Propose the LockStep Broadcast Tree (LSBT) to solve the big-data broadcasting problem over heterogenous networks by constructing an efficient pipeline broadcasting tree. We contribute to the understanding and investigation of how to design a scalable and practical algorithm to compute an LSBT such that its maximum completion time is minimized.

2) Design a novel polynomial-time algorithm to select the optimal uplink rate r^* for building an optimal LSBT. To the best of our knowledge, this work is the first study to investigate the relation between a single overlay tree (with a fixed uplink rate r^*) and the broadcast operation based on uplink sharing model. Unlike the previous works [13], [14] in which the criteria is to maximize system throughput, to the best of our knowledge, this is the first to study the problem of designing a tree overlay network aiming at minimizing the maximum completion time.

3) Introduce several original applications based on the LSBT model. Specifically, given a data delivery deadline, one can estimate whether a delivery job through a specific network could meet its deadline based on the LSBT model. Developers may take advantage of this property to can maximize the performance of collaborative applications in datacenter networks.

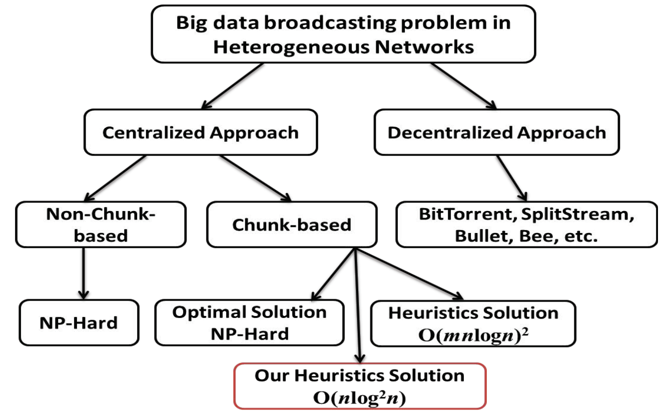

Big data broadcasting problem in Heterogeneous Networks
Centralized Approach
Decentralized Approach
Non-Chunk-based
Chunk-based
BitTorrent, SplitStream, Bullet, Bee, etc.
NP-Hard
Optimal Solution NP-Hard
Heuristics Solution O(mnlogn)^2
Our Heuristics Solution O(nlog^2n)

Fig. 1. Scope of our contributions

The rest of this paper is organized as follows. In the Section 2, we give background of the big data broadcasting problem. We state the general big data broadcasting problem and introduce our LSBT model and its applications in the Section 3. The detail of our optimal algorithm for LSBT problem is presented in Section 4. Numerical evaluation are presented in Section 5. The Section 6 describes the context of related work. We conclude this paper and present ideas for future research in Section 7.

2 Background

Suppose that m data chunks of equal size are initially held by a single source node in a network. The data broadcasting problem is about disseminating these m chunks to a population of n nodes in as less time as possible, subject to the uploading link capacity constraints of nodes. This problem has been studied in the context of many different network scenarios, such as homogenous and heterogenous networks. For interested readers, a comprehensive survey can be found in the article [15]. In this paper we focus on big data broadcasting problem in heterogenous networks. Figure 1 illustrates these solutions to the big data broadcasting problem in heterogenous networks along multiple dimensions.

For the centralized approach, we first look at the results of the Non-Chunk based approach. Khuller and Kim [11] showed that the problem of minimizing the completion time for broadcasting a single chunk (a message) in heterogenous networks in a NP-hard problem. The authors also showed the Fastest-Node-First (FNF) heuristic method gets a performance ratio of at most 1.5 and the FNF results in optimal solutions in many cases for single chunk broadcast. In additional, Liu [16] showed that the FNF heuristic method is optimal in only two classes of nodes. However, the data broadcasting problem is more complicated when the data consists of multiple chunks and it is still an open problem: Can data broadcasting problem with multiple chunks be solved by a polynomial time algorithm? [17].

Within the Chunk-based methods, the optimal solution has been shown in the article [12]. The authors presented an uplink-sharing model for the well-known data broadcasting problem and formulated data broadcasting problem as a mixed integer linear programming (MILP). However, as the numbers of variables in the linear programming grows exponentially n and m, this method is not practical for large n and m. Goetzmann *et. al.* [18] show that if peer capacities are heterogeneous and symmetric, this problem becomes strongly NP-hard. A recent result [19] presented two heuristic algorithms to schedule data chunks transfer between nodes. The time complexity of both two centralized algorithms is O$(m \times nlogn)^2$.

For the decentralized approach, many decentralized systems have been proposed to disseminate chunks via an overlay topology. With overlay-based approaches, nodes maintain a set of overlay links to other nodes and exchange chunks among neighboring nodes. BitTorrent [20], SplitSteam [21], Bullet [22] and Bee [23] are some examples of the overlay-based approach. In [23], the authors showed Bee can approach lower bound of the maximum completion time in heterogenous networks by simulations. In this paper, we retain the interest in the centralized approaches, thus interested readers can find a comprehensive survey of these decentralized systems in the article [24].

3 Problem Statement

The assumption in our model is similar to the Uplink-Sharing model proposed by J. Mundinger *et al.* [12]. Each node can simultaneously connect to other nodes and the available upload capacity of a link is shared equally amongst the uploading connections. Based on the Uplink-Sharing model, we model the nodes and data transfer networks as the nodes and edges of a direct graph. We assume that there are n nodes in a network system, denoted by n_1, n_2, n_3,..., n_n, where the broadcasting source is node n_1 and the $n-1$ nodes have upload capacities $c = \{c_1, c_2, c_3, \ldots, c_n\}$, measured in kilobyte per second (KBps). Besides, we also assume that the source node, n_1, has the data item that is divided into m chunks of equal size, to disseminate to all the other nodes, and c_1 is larger than or equal to that of other nodes. Finally, we assume that the downloading capacity of each node is larger or equal to its uploading capacity. This is true for virtually all existing network access technologies, e.g., ADSL or cable modems.

3.1 LockStep Broadcast Tree (LSBT) problem

To reduce the complexity of the original data broadcasting problem [11], [12], we model it as the LockStep Broadcast Tree (LSBT) problem. By this we define a performance goal for a single LSBT, that is achieving minimum completion time by optimizing the basic bandwidth allocation, r, among LSBT nodes. Different from original problem, we allow data be divided into chunks and sent in a pipeline fashion. Formally, given a set of n nodes $\mathcal{N} = \{n_1, \ldots, n_n\}$, each node n_i is connected to the network via an access link of upload capacities c_i and a size of chunks B. The LSBT problem is to determine the upload bandwidth r^* of each uplink to build the LSBT t, in which node n_i should allocate upload bandwidth r^* to each connection to its child nodes in order to minimize the maximum completion time D for propagating a data chunk. Note that it is possible to handle simultaneously several connections and to fix the bandwidth allocated to each connection [25]. In the following definition, we define the number of edges k in each node for LSBT.

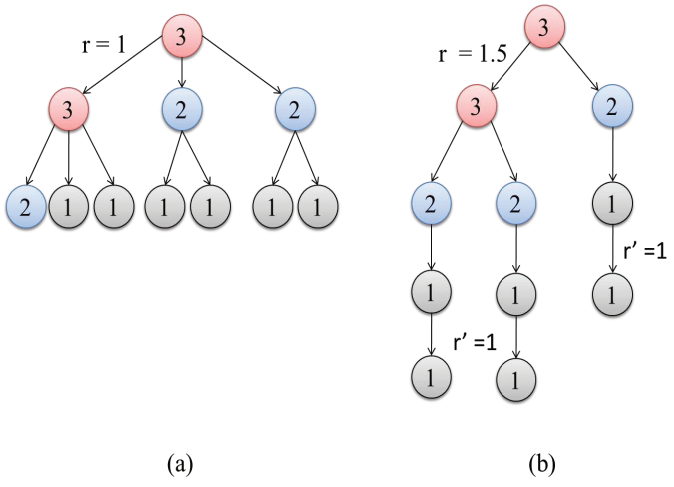


Fig. 2. The two examples of LSBT. The tree (a) presents the optimal LSBT with r^* = 1, the maximum completion time D is 2 units of time and tree (b) requires 3 units of time. Assume that the size of data chunk $B = 1$ and the digits specify these node's upload capacity.

Definition 1. *For each LSBT node n_i, the number of edges (uploading connections) k_i is depended on its upload capacity, i.e., $k_i = \lfloor \frac{c_i}{r} \rfloor$, for $1 \le i \le n$ and $\forall\ r \in \mathbb{R}^+$.*

The formal mathematic definition of the maximum completion time D is shown as follows.

$$r^* = \arg\min_{r \in \mathbb{R}^+} D(c, r) = \arg\min_{r \in \mathbb{R}^+} \sum^{h(t^{(c,r)})} \frac{B}{r}, \tag{1}$$

where $t^{(c,r)}$ is the LSBT with the set of upload capacity c and an upload bandwidth r, $h(t^{(c,r)})$ describes the function that returns the height of the LSBT $t^{(c,r)}$.

Note that this general Equation (1) removes restrictions on the location of nodes in the network, it only calculates the propagation delay of data chunks from the root to the leaves. Moreover, LSBT model addresses the data broadcasting problem by building a single broadcast tree, in which nodes can transmit data chunks in a pipeline manner. Thus the maximum completion time D is the summation of the transmission time of a data chunk (*i.e.*, $\frac{B}{r}$) in each level of the LSBT $t^{(c,r)}$.

Example. Figure 2 shows the two examples. Given a set of eleven nodes having upload capacities {3, 3, 2, 2, 2, 1, 1, 1, 1, 1, 1}, we can build at most 11^{11-2} different broadcasting trees (by Cayley's formula [26]). However, there exists an optimal LSBT constructed by sorting the nodes according to their number of edges in non-increasing order. We will show this important property of LSBT in the next section (**Theorem** 3). In Figure 2, the tree (a) presents the optimal LSBT t^a with upload bandwidth $r^* = 1$. Here, we assume that the size of data chunk $B = 1$. Since $h(t^a) = 2$, the maximum completion time D of tree t^a should be 2 (*i.e.*, $\frac{1}{1} + \frac{1}{1} = 2$) units of time (by Equation 1). The other tree t^b in Figure 2 is not an optimal LSBT. The maximum completion time D of tree t^b is 3 units of time (*i.e.*, $\frac{1}{1.5}+\frac{1}{1.5}+\frac{1}{1.5}+1=3$). Note that in tree t^b these gray nodes in the 3^{th} level only can provide one unit of upload capacity to their child nodes even if r is specified as 1.5.

3.2 Potential Applications of LSBT

We envision that our LSBT could be well-suited for a host of applications. There are at least three broad applications where LSBT can be applied: 1) topology control in BitTorrent-like systems; 2) data broadcasting in cloud computing software stack; 3) energy conservation in peer-assisted content delivery services. We consider these in the context of network systems that are heterogenous network environments.

First, Our algorithm of LSBT could be useful in topology control in BitTorrent-like systems [20]. BitTorrent is a peer-to-peer application that aims to enable the fast and efficient distribution of large files among a large group of nodes. In BitTorrent, each peer maintains a constant number of concurrent upload connections (usually five). Please see the article [20] for more detailed descriptions. Recent studies [23], [27], [28] show that the fixed upload connections limit is harmful to uplink utilization and peer fairness in BitTorrent. However, how to decide an appropriate number of concurrent uploads in BitTorrent still is a challenge. The proposed algorithm for LSBT may provide an insight into selecting the number of concurrent uploads in BitTorrent-like systems.

Second, LSBT can be integrated into the cloud computing software stack. For example, Apache Hadoop[1] is a software framework that allows for the distributed processing of large data sets across clusters with thousands machines. Thus an efficient and scalable way to disseminate a large volume of data among machines is a significant challenge in Hadoop [8]. Another example is the delivery services of OpenStack[2], it is designed for virtual appliance deployment in datacenters. Our LSBT can be integrated into the delivery services of OpenStack software stack. A number of algorithms and protocols have been proposed, implemented, and studied [8], [29]. For any data delivery job initiated by cloud computing softwares, there is an associated deadline. The main advantage of LSBT is to enable these cloud software stacks to predict and schedule the associated deadline of a data delivery job. Specifically, given a data delivery deadline, LSBT may be possible to determine that can the network system meet the deadline or what is the possible deadline for the delivery job. This advantage can severely impact application performance in datacenter networks.

For example, the police office of New York City attempts to build a public street surveillance system with many thousands of cameras. The design should include a distributed datacenter architecture to store and to process the large-scale video streams. These street videos are kept in order to allow retrieval and review in the event a crime was committed. The success of the design relies on the cooperation and elastic resource utilization of these multiple distributed datacenters. Thus when a crime event was committed, the related video streams should be disseminated from the hosted datacenter to other datacenters for rapidly computation. LSBT may be possible to enable software stacks in the distributed datacenter architecture to predict and schedule the associated deadline of a data delivery job. It may bring the benefits in terms of elastic resource utilization and managing delivery of the computational jobs.

Finally, Our algorithm for LSBT could be useful to answer the question: what is the maximum streaming rate that can be sustained for all receivers within a peer-assisted content delivery service provider. Many content delivery service providers, such as PPLive[3], Akamai[4], that may rely on participating users contributing uplink bandwidth to scale up delivery services to hundreds of thousands of users. However, if the total contributed bandwidth from the service provider and participating users can not support to the demanded quality of services (*ex.*, H.264[5]/768kbps), the service provider should increase contributed bandwidth (servers) from server-side. For energy conservation and environmental issues, it is an interesting and significant issue to investigate how to dynamically increase or decrease the number of servers in accordance with the demanded QoS and the number of active users. Given a set of node upload capacities c, our LSBT algorithm can roughly sketch out the coarse-grained QoS level (*i.e.*, r^* Kbps) of the current system and be used to regulate the energy consumption in server-side. Thus, our LSBT model can be used for creating a systematic approach that arranges server-side resources for peer-assisted content delivery protocols. To the best of our knowledge, little work [30] has been conducted on energy conservation in peer-assisted content delivery services. In future work, we aim to apply our LSBT algorithm to the research direction.

4 OPTIMAL LOCKSTEP BROADCAST TREE

In this section, we present our LSBT algorithm that is also a heuristic for the data broadcasting problem. Given a set of node upload capacities c, we aim at finding an optimal LSBT, that is a data broadcast tree where data chunks can be sent in a pipelined manner. We provide a thorough analysis of LSBT in both homogenous and heterogenous network systems. We first clarify LSBT in homogenous networks cases and describe the LSBT algorithm in heterogenous network cases later.

4.1 Homogenous Network Systems

We present the optimal solution of LSBT when the upload capacities of nodes are identical. In general, we assume that all nodes have upload capacity of c. Mundinger *et al.* [12] have presented the optimal scheduling solution for broadcasting multiple messages on the uplink-sharing model. The following **Theorem** 1 (***Mundinger's theorem***) is proved in the article [12]. If each round costs one unit of time, then the maximum completion time of the optimal solution is $m+\lfloor\log_2 n\rfloor$, where m is the number of chunks and n the

1. http://hadoop.apache.org/
2. http://openstack.org/
3. http://www.pptv.com/
4. http://www.akamai.com/
5. http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

number of nodes. Note that each node can only upload one data chunk to another node in each round. By contrast, each node can send a data chunk to k other nodes simultaneously in the LSBT model.

Theorem 1. *(Mundinger's theorem [12]) In homogenous network systems, the minimum number of rounds required to complete the broadcasting of all data chunks is $m + \lfloor \log_2 n \rfloor$, where m is the number of data chunks and n is the number of nodes.*

In our LSBT model, the maximum completion time D is equal to Equation (1). However, due to the upload capacities of all nodes are equal, it can be simply expressed as follows (note that $r = \frac{c}{k}$).

$$\begin{aligned} D &= \frac{B}{r} \log_k n \\ &= \frac{kB}{c} \frac{\ln n}{\ln k}, \end{aligned}$$

where B is a size of data chunks.
Let $\mathcal{G} = \frac{B \ln n}{c}$, we have

$$D = \mathcal{G} \frac{k}{\ln k}. \tag{2}$$

Set $\frac{dD}{dk} = 0$,

$$\frac{dD}{dk} = \frac{\mathcal{G}}{\ln k} - \frac{\mathcal{G}}{(\ln k)^2} = 0. \tag{3}$$

The Equation (3) implies that

$$\begin{aligned} \ln k &= 1, \\ k &= e. \end{aligned} \tag{4}$$

It can be shown that Equ. 2 is a convex function. Thus we have the following theorem in discrete model.

Theorem 2. *In homogenous network systems, the optimal value r^* for LSBT is either $c/2$ or $c/3$ that makes the LSBT minimize the maximum completion time, where c is the upload capacity of all nodes.*

Figure 3 illustrates a simple numerical example of LSBT in a homogenous network, in which we set $n = 100$, $c = 1$, and $B = 1$. We then calculated the maximum completion time D in Equation (2). In the results, all nodes have k upload connections, the value of k depending on the considered scenario. We can see that the numerical results significantly depend on the value of k in homogenous network systems, and the LSBT can minimize the maximum completion time when k is equal to e as we shown in Equ. 4 in continuous model.

4.2 Heterogenous Network Systems

We now consider the general LSTB model in which nodes' upload capacities may be different. First, we present an algorithm to construct an optimal LSBT for a given rate r. We then give both the upper and lower bounds of the value of r^*. Finally we present an $O(n \log^2 n)$ algorithm to select the optimal upload bandwidth r^* of each uplink and to construct the optimal LSBT.

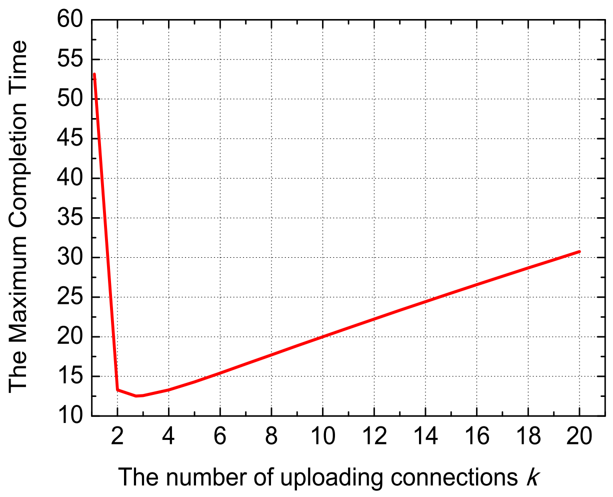


Fig. 3. Numerical results of LSBT in homogenous networks. Assume that the number of nodes $n = 100$, the size of data chunk $B = 1$, and the upload capacity of all nodes $c = 1$.

We now present the algorithm GLSBT to generate an LSBT t which is shown to be optimal for the given rate r. Given a set of nodes $\mathcal{N} = \{n_1, n_2, \cdots, n_n\}$ with c_i as the upload capacity of node $i, 1 \leq i \leq n$, and a real number r to denote the rate of the LSBT. We assume that the nodes are given in non-increasing order of their upload capacity, *i.e.*, $c_j \leq c_i$ if $i < j$. In the algorithm GLSBT, given r as the rate of the LSBT, the degree of each node n_i is given by the line 4. The algorithm GLSBT generates the LSBT t by assigning a node n_q as the parent of the node n_l if and only if q is the smallest integer such that $l \leq 1 + \sum_{i=1}^{q} k_i$, where $1 \leq l \leq n$. It is easy to show that the time complexity of the algorithm GLSBT is $O(n)$, where n is the number of nodes. Thus we have the following lemma.

Lemma 1. *The algorithm GLSBT (Algorithm 1) can be made to run in O(n) time in a network system of size n.*

Then we present the following theorem to show that the algorithm GLSBT gives an optimal LSBT for the given rate r.

Theorem 3. *Given an uplink rate r^*, building the LSBT t that is constructed in a way that any child node's out-degree is always less than or equal to its parent's and providing that t is optimal in terms of the maximum completion time D.*

Proof: Suppose that we have a set of n nodes, and $k_1, k_2, k_3, \ldots, k_n$ are the umber of edges of each node. Then by **Definition** 1: $k_i = \lfloor \frac{c_i}{r} \rfloor$, for $1 \leq i \leq n$. Let $k_1, k_2, k_3, \ldots, k_p$ be the edges of nodes in a LSBT as shown as Figure 4, where p is the smallest integer such that $\sum_1^p k_p \geq (n-1)$. There are two cases impact on the height of LSBT.

Case I) Assume that an optimal LSBT t is constructed in the order of $(k_1, k_2, k_i, \ldots, k_j, k_p)$ and $k_i < k_j$, for $1 \leq i < j \leq p$. Let h be the height of t. If k_i and k_j are interchanged, then there exists another optimal LSBT t' with height h' in order by $(k_1, k_2, k_j, \ldots, k_i, k_p)$. Since $k_j > k_i$, h' is less than or

Algorithm 1 GLSBT: An algorithm for generating a LSBT t

```
Input: a set of upload capacities $c$ and a rate $r$ for LSBT $t$
Output: a LSBT $t$
1:  BEGIN
2:  /* Computing the degree of the node $n_i$ */
3:  for $i \leftarrow 1$ to $n$ do
4:      $LSBTree[i].Degree \leftarrow \lfloor c_i/r \rfloor$
5:  end for
6:  $NodeIndex \leftarrow 1$
7:  $NodeCount \leftarrow 2$
8:  while $NodeCount \leq n$ do
9:      $k \leftarrow LSBTree[NodeIndex].Degree$
10:     for $j \leftarrow 1$ to $k$ do
11:         $LSBTree[NodeCount].parent \leftarrow NodeIndex$
12:         $NodeCount = NodeCount + 1$
13:     end for
14:     $NodeIndex = NodeIndex + 1$
15: end while
16: return $LSBTree[]$
17: END
```

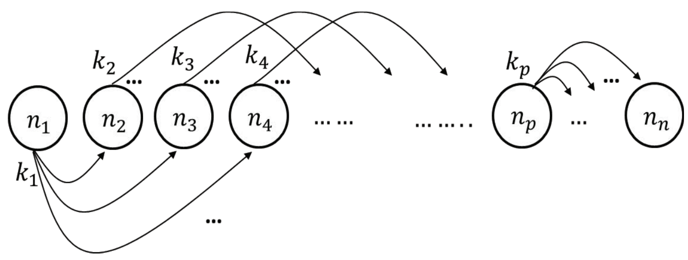


Fig. 4. An illustration of generating a LSBT

equal to h. Therefore t' is an optimal LSBT (*i.e.*, $\frac{Bh'}{r^*} \leq \frac{Bh}{r^*}$).

Case II) Assume that an optimal LSBT t is created by the order $(k_1, k_2, k_i, \ldots, k_p, k_j)$ and $k_i < k_j$, for $1 \leq i < j \leq n$. Similarly in switching k_j and k_i, we get the new LSBT t'. Since $k_j > k_i$, the height of t' is also less than or equal to the height of t. Therefore t' is an optimal LSBT. □

Then we provide lower bound and upper bound of the value of r^* as follows.

Lemma 2. *(Lower bound) In heterogenous network systems, the lower bound of r^* in the optimal LSBT is larger than or equal to $\frac{c_1}{n-1}$, where $c_1 \geq c_i$ for $1 < i \leq n$.*

Proof: If this is not true (*i.e.*, $r^* < \frac{c_1}{n-1}$), then we have the optimal LSBT t' where $r' < \frac{c_1}{n-1}$. There exists another LSBT t, where $h(t) = 1$. The value of r^* in t is equal to $\frac{c_1}{n-1}$ and the value of D in t is equal to $\frac{B(n-1)}{c_1}$. However, the value of D in t' is larger than $\frac{B(n-1)}{c_1}$. This contradicts the assumption that $r^* < \frac{c_1}{n-1}$. □

Lemma 3. *(Upper bound) In heterogenous network systems, the upper bound of r^* in the optimal LSBT is less than $\frac{\sum c_i}{n-1}$, for $1 \leq i \leq n$.*

Proof: A tree has n vertices and $n-1$ edges. It implies

$$r^* \times (n-1) \leq \sum_{i=1}^{n} c_i.$$

However, the leaf nodes in LSBT can not contributes their upload capacities, thus

$$r^* \leq \frac{\sum_{i=1}^{(n-l)} c_i}{n-1} < \frac{\sum_{i=1}^{n} c_i}{n-1},$$

where l is the number of the leaf nodes in a LSBT. □

Next, we give the details of the algorithm for the selection of r^*. As described in Equation 1, $r \in \mathbb{R}^+$, so it means the possible value of r is infinite, even both the upper and lower bounds of the value of r^* are given. Since the number of r is infinite, an efficient discretization algorithm of r is critical. In LSBT, we propose a simple division algorithm to discretize the value of r. This algorithm comes from the observation: the number of upload connection (*i.e.*, k) in each LSBT node is a positive integer and $1 \leq k \leq (n-1)$. Thus we enumerate all possible candidates of r^* which make k an integer. Algorithm 2 presents our solution to discretize the value of r. Let $CandidateSet$ denote the set of the possible value of r^*, and the binary search will be performed on it.

In Algorithm 2, it first reduces the redundance of c_i by preforming an union operation (named $UnionSet$) of each c_i, for $1 \leq i \leq n$ and sorting the set (in line 4-7). Next, the loop from line 8 to 18 is used to discretize the value of r and to filter out the extreme r values restricted by the upper and lower bounds. In the loop, it gets candidates of r by computing $u/k, \forall\ u \in UnionSet$ and $1 \leq k \leq (n-1)$, and puts those candidates into the $CandidateSet$. Note that the number of candidates is O(n^2) if each LSBT node has an unique upload capacity. However, the filter scheme can significantly reduce the number of candidates. We will show the experimental results in the next section.

Before we present the binary search algorithm for selecting the value of r^*, we first show the following lemma and theorem which provide properties to derive the efficient binary search algorithm on r^*.

Lemma 4. *Given the discrete spectrum of r for building a LSBT t, the value of r^* occurs in one of the values that change the height of t.*

Proof: Suppose that the lemma is not true, there exists an optimal LBST $\hat{t}$ built with the value of $\hat{r}$, its height is h, and the next value of $\hat{r}$ in the discrete spectrum (labeled as r' and $r' = \hat{r} + \delta,\ \delta > 0$) does not increase the height of $\hat{t}$. According to Equation (1) and $r' > \hat{r}$, there is another LSBT t', its height is h, and the maximum completion time of t' is less than the one of $\hat{t}$. This contradicts the assumption that $\hat{t}$ is an optimal LSBT of height h. □

Lemma 5. *The height of any rooted tree with n nodes must be less than $\log_2$ n if the out-degree of every internal node is greater than 1.*

Proof: We prove it by contradiction. Assume there exists a tree, t, with n nodes having the height greater than $log_2 n$

Algorithm 2 A discretization algorithm for the candidateset

```
Input: a set of upload capacities $c$ and the upper and lower
    bounds of $r^*$
Output: $CandidateSet$
 1: BEGIN
 2: $UnionSet \leftarrow empty$
 3: $CandidateSet \leftarrow empty$
 4: for $i \leftarrow 1$ to $n$ do
 5:     $UnionSet \leftarrow UnionSet \cup c_i$
 6: end for
 7: $UnionSet \leftarrow Sort(UnionSet)$
 8: for $k \leftarrow 1$ to $n-1$ do
 9:     for all $u$ in $UnionSet$ do
10:         $r \leftarrow u/k$
11:         if $r \geq$ upper then
12:             continue
13:         else if $r <$ lower then
14:             break
15:         end if
16:         $CandidateSet \leftarrow CandidateSet \cup r$
17:     end for
18: end for
19: return $CandidateSet$
20: END
```

while all internal nodes in t have out-degree greater than 1. Given t's height greater than $log_2 n$, we get

$$h > log_2 n \Rightarrow n < 2^h. \tag{5}$$

We now count the number of nodes in each level of t. Because every internal node has out-degree greater than 1, at level i there are at least 2^i nodes.

$$\begin{aligned} n &\geq 2^0 + 2^1 + 2^2 + \cdots + 2^{h-1} + c \\ &\geq 2^h - 1 + c, \end{aligned} \tag{6}$$

where $1 \leq c \leq 2^h$. Please note that the root node is at level 0 and the c is the number of nodes at the last level. There is no such n fitting both Eq. 5 and Eq. 6, so that no such tree t exists. □

Theorem 4. *The height of any optimal Lock-Step Broadcast Tree (LSBT) with n nodes is less than or equal to $2 \times log_2 n$, where n is the number of nodes.*

Proof: We prove it by contradiction. We assume there is a optimal LSBT, t, with n nodes and its height is greater than $2 \times log_2 n$.
By the Lemma 5, we know that in t the out-degree of some internal nodes must be equal to 1 (*i.e.*, less than 2). We make a tree, t', by setting the rate $r' = \frac{r}{2}$, where r is the rate for t. Since the out-degree of all internal nodes in t' must be greater than or equal to 2, the height of t' must be less than $log_2 n$ by the Lemma 5.

The completion time for t' will be

$$D(t') = \frac{h'}{r'} \leq \frac{log_2 n}{\frac{r}{2}} = \frac{2 \times log_2 n}{r}.$$

Algorithm 3 The r^* search algorithm for the optimal LSBT

```
Input: a set of upload capacities $c$ and $CandidateSet$
Output: $r^*$ and $D^*$
 1: BEGIN
 2: $CandidateSet \leftarrow Sort(CandidateSet)$
 3: $D^* \leftarrow \infty$
 4: $r^* \leftarrow empty$
 5: for $h \leftarrow 1$ to $2(\lfloor \log_2 n \rfloor + 1)$ do
 6:     $right \leftarrow 1$
 7:     $left \leftarrow Sizeof(CandidateSet)$
 8:     while $right \leq left$ do
 9:         $mid \leftarrow \lfloor (left + (right - left)/2) \rfloor$
10:         $r \leftarrow CandidateSet[mid]$
11:         $t \leftarrow GLSBT(c, r)$
12:         if $left - right = 1$ and $t.Height = h$ then
13:             $d \leftarrow t.BroadcastingTime$
14:             if $d < D^*$ then
15:                 $r^* \leftarrow r$
16:                 $D^* \leftarrow d$
17:             end if
18:         else
19:             break
20:         end if
21:         if $h \leq t.Height$ then
22:             $right \leftarrow mid$
23:         else
24:             $left \leftarrow mid - 1$
25:         end if
26:     end while
27: end for
28: return $r^*$ and $D^*$
29: END
```

Given $D(t) = \frac{h'}{r'} > \frac{2 \times log_2 n}{r}$, we got $D(t') < D(t)$ which contradicts that t is an optimal LSBT. □

Algorithm 3 describes our scheme to search the value of r^* to build an optimal LSBT. Searching the r^* is much like searching a binary search tree, except that instead of searching the value of r, it make a seeking condition both on the value of r and the height of LSBT h. Algorithm 3 takes as input a set of upload capacities c and $CandidateSet$ obtained by Algorithm 2.

First, Algorithm 3 takes O($nlogn$) time to sort the $CandidateSet$ in the line 2. Then for each different height of LSBT (in line 5-27), it searches the optimal value of r^* and returned the best value of r^* and the maximum completion time (in line 28). The value of h is restricted to $2(\lceil log_2 n \rceil + 1)$ (in line 5) because of **Theorem** 4. Thus we can only check the height of LSBT from 1 to $2(\lfloor \log_2 n \rfloor + 1)$, it runs in O($log_2 n$) time on a LSBT tree of height $2(\lfloor \log_2 n \rfloor + 1)$. During the loop (in line 8-26), Algorithm 3 performs a straightforward generalization of the binary searching procedure, it takes O($log_2 n^2$) time. (Note that n^2 is the size of the candidate set.) In line 11, $GLSBT()$ is an O(n) function for building a LSBT according to a specified r and it returns the LSBT t (by **Lemma** 1). Lines 13-19 check to see if we have now discovered the value of r^* for the specified h, and update

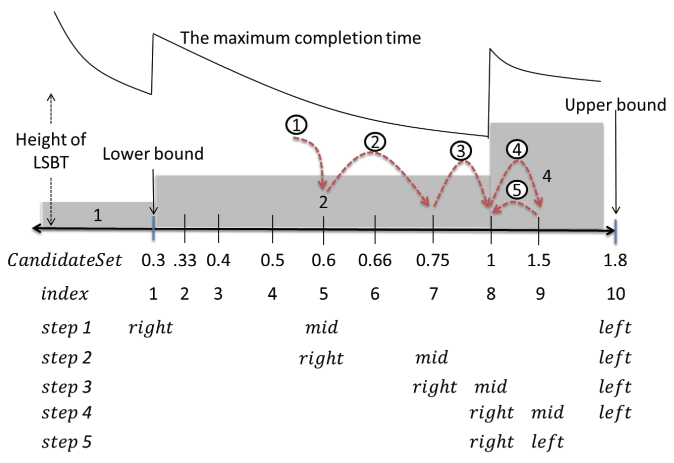


Fig. 5. An illustration of the binary search algorithm for selecting the value of r^* in the LSBT of height $h = 2$

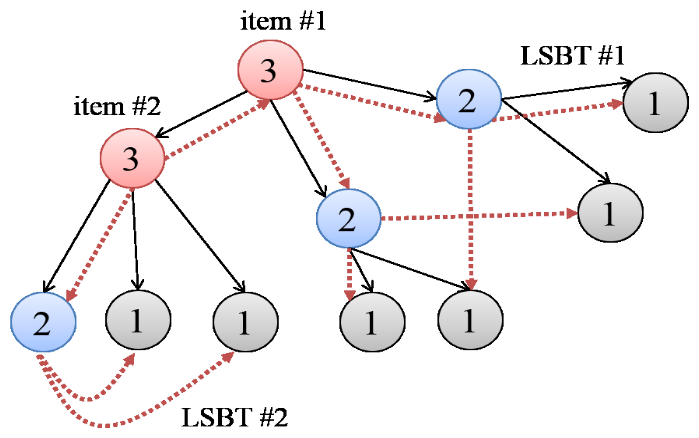


Fig. 6. An example of LSBT for multiple data sources

the best r^* and D^* if we have. Note that by **Lemma** 4, the line 12 presents the successful condition for searching. The line 19 terminates the search unsuccessfully, *i.e.*, an optimal LSBT of height h does not exist. In summary, the procedure runs in O($2log_2n \times log_2n^2 \times n$) time, thus we have proved the following theorem.

Theorem 5. *The r^* search algorithm (Algorithm 3) for an optimal LSBT can be made to run in O($(nlog^2n)$) time on a set of upload capacities c.*

Theorem 6. *The r^* search algorithm (Algorithm 3) correctly computes an optimal LSBT of any set of upload capacities c.*

Our proof of **Theorem** 6 is based on the following observation.

Observation 1. *Given a set of upload capacities c, the Algorithm 2 shall generate the all candidates of possible r^*. Because of the number of upload connection (i.e., k) in each LSBT node is a positive integer and $1 \leq k \leq (n-1)$, where n is the number of nodes.*

The proof of **Observation 1** is a straight-forward inductive argument on the value of k, $1 \leq k \leq (n-1)$.

Proof: (**Theorem** 6) The r^* search algorithm (Algorithm 3) clearly finishes in a finite number of steps. The number of executions of the for-loop (line 5-27) is restricted to $2(\lceil log_2n \rceil + 1)$ by **Theorem** 4.

For each height of LSBT trees (in while-loop, line 8-26), it searches the optimal value of r for the LSBT t's height $= h$ in the all possible candidates of r^* based on **Observation 1**.

During the body of the for-loop (line 5-27), it keeps the best value of r^* and the maximum completion time of an optimal LSBT tree (line 14-17). This statement is easily seen to be true.

□

Now we give an example of the r^* search algorithm (Algorithm 3) as follows.

Example. Figure 5 shows the illustration of searching results by applying Algorithm 3 to the example in Figure 2. The numerals in these gray areas in Figure 5 mean the heights of these LSBTs constructed by each different value of r. Recall that the given set of eleven nodes come with a set of upload capacities $\{3, 3, 2, 2, 2, 1, 1, 1, 1, 1, 1\}$ in Figure 2. The numbered dashed lines specify the series of steps of the searching operation. According to **Lemma** 2 and **Lemma** 3, we can obtain the lower bound (*i.e.,* 0.3) and upper bound (*i.e.,* 1.8) in the example of Figure 2. As shown in Figure 5, the $CandidateSet$ consists of ten elements that returned by Algorithm 2. The points on horizontal axis presents the spectrum of the possible value of r after the sorting operation, and the topmost curve specifies the maximum completion time D procured by individual values of r. In the running example, we consider $h = 2$, *i.e.*, selecting the value of r^* for the optimal LSBT with height 2. The bottom lines in Figure 5 illustrates the variances of $right$, $left$ and mid in each step. The total number of step during the searching operation is five and the searching operation is terminated when $mid = right = 8$ and $left = 9$ (line 13 in Algorithm 3).

Finally, we show that the maximum number of rounds required to complete the broadcast of entire data chunks in an optimal LSBT is O($m + logn$) steps. The proof of **Theorem** 7 requires to combine **Theorem** 2 and **Theorem** 4.

Theorem 7. *In an optimal LSBT, the minimum number of steps required to complete the broadcast of entire data chunks is O($m + logn$), where m is the number of data chunks of equal size and n presents the number of nodes in a network system.*

4.3 Multiple Data Sources

A number of issues require further consideration. Specifically, in big data computing, data is typically partitioned among multiple sources. Our current LSBT algorithm does not address the partitioned data problem, we discuss the possible solution as follows.

One of the most straightforward way to apply the current LSBT algorithm to a partitioned data sources environment is to build multiple LSBT broadcast trees for each data item. For example, as shown in the Figure 6, there are two data items, *i.g.*, $item\#1$ and $item\#2$, in the multiple data sources environment. In this example, we can build two LSBT broadcast trees for those data items easily to broadcast multiple data sources. Note that the time complexity of our LSBT building algorithm is O($(nlog^2 n)$), it should be efficient and effective.

5 Numerical Evaluation

In this section, we analyze the performance of our LSBT through numerical evaluations. The algorithm developed in this paper can be embodied in the control plane of big data service stacks to form node relationships that achieve the capacity. In the numerical results, we have implemented three approaches including FNF heuristic [11], DIM-Rank heuristic [19] and our LSBT. All of above heuristic algorithms are centralized, however in DIM-Rank, the cost of computing the broadcast schedule is non-trivial. Note that DIM-Rank is the best algorithm in [19] by comparing other state-of-the-art algorithms. The node' uplink capacities distribution is set according to the actual Internet that is reported in [31] and their respective fractions in the node population are summarized in Table 1.

5.1 The Details of Compared Techniques

In Section 2, we briefly discussed prior techniques for broadcasting big data in heterogeneous networks. We discuss here these works in more detail.

5.1.1 FNF heuristic

The Fastest-Node-First (FNF) heuristic technique [32] is a common centralized algorithm used to find a broadcast tree in heterogeneous networks. The algorithm is simple and easy to implement. We briefly describe it as follows: in each iteration of FNF heuristic, it selects a sender form the set of nodes that have received the message and a receiver which has not received the message. Obviously at each iteration, FNF heuristic needs to make two decisions. First, it has to decide which sender is going to send the message to the new receiver. The second decision is to choose the new receiver among the nodes which have not been added to the tree yet. The intuition of FNF heuristic is that always picking the fastest sender and the fastest receiver is the best way to deliver the message quickly. By this way, FNF heuristic can generate a greedy tree through which broadcast operations can be implemented. Note that the FNF heuristic algorithm is restricted to one message. Thus we use it to broadcast multiple chunks one by one.

5.1.2 DIM-Rank

The DIM-Rank technique is a centralized heuristics algorithm for multi-chunk dissemination in heterogenous networks. In DIM-Rank, a centralized scheduler is used to arrange data chunk delivery between nodes based on the pair-wise bandwidth and latency measured among all participating nodes. In other hand, the scheduler is used to make decisions on the selections of senders, receivers and chunks at each iteration. In DIM-Rank, two schedule metrics are developed to help scheduler make decisions. First, **Chunk Spread** is the total number of nodes that have the same data chunk. Second, **Node Rank** is the summed inverse of spread of all data chunks that a node contains, *e.g.*, if a node contains rare data chunks or many chunks then it's rank is higher. At each iteration, DIM-Rank technique sorts nodes first by their node rank from lowest to highest, and then sorts nodes by their upload capacity (highest to lowest). After sorting, the scheduler picks receivers from the sorted list for each sender, and the lowest-spread chunk is chosen and transmitted. For more details, please refer to the article [19].

TABLE 1
Node Uplink Capacity Distribution

Uplink Distribution				
Uplink Capacity (Kbps)	128	384	1000	5000
Fraction (%)	20	40	25	15

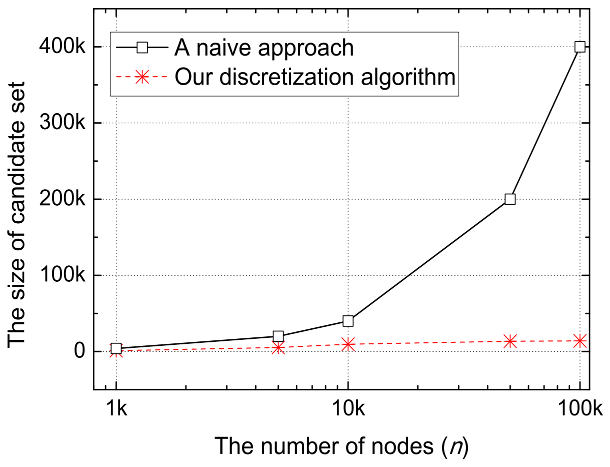
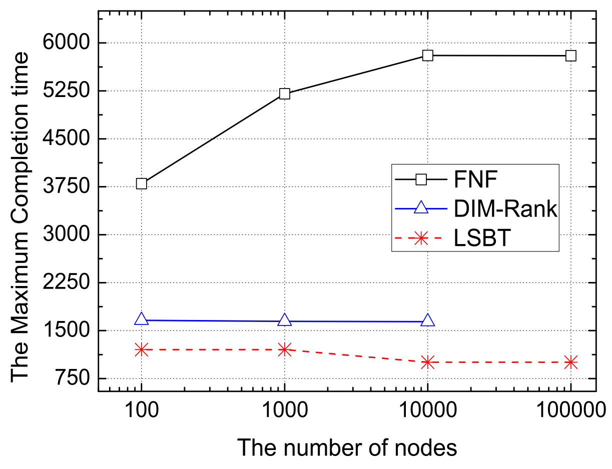


Fig. 7. The size of candidateset versus the number of nodes n

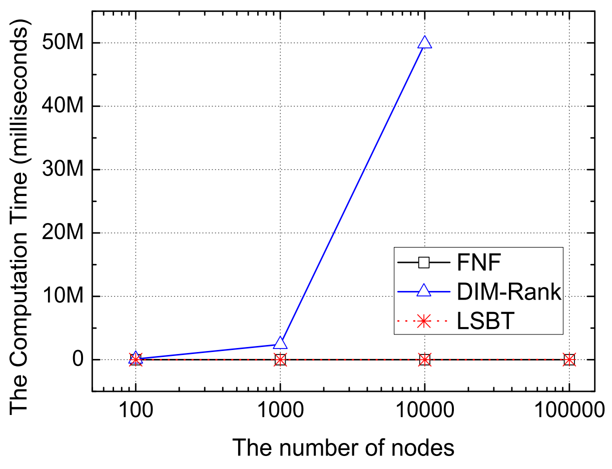
5.2 Experimental Results

We first study the size of candidate set that derived from the Algorithm 2. Figure 7 shows the effect when the total number of nodes to be broadcasted is increased. Note that the x-axis is also a log-scale ($log_{10}n$). Algorithm 2 has a clear superior performance over the naive approach. Moreover, as the number of node is increased, the gap widens between Algorithm 2 and the naive approach making it very desirable. Intuitively, in the naive approach, the worst case of the size of $CandidateSet$ is the number of nodes multiplied by the size of $UnionSet$. Thus, the solution of Algorithm 2 may give a good heuristics for reducing the size of $CandidateSet$ in a large scale network.

We now show the maximum completion time of the three algorithms under various scenarios. We consider networks with n= 100, 1000, 10000 and 100000 nodes. The size of file is 100MB and the number of data chunks is 1000. Figure 8(a) shows the total time each algorithm taking to broadcast the file to all the nodes. Note that the x-axis is a log-scale of number of nodes and thus a straight line indicates good scalability, such as log-scale ($\log_{10} n$). Figure 8(b) shows the computation time of each algorithm to schedule the broadcast job. By the simulation results, LSBT performs the best while



(a) The Maximum Completion time (the unit on y-axis is second)



(b) The Computation Time (the unit on y-axis is million milliseconds)

Fig. 8. Performance comparison with increasing the number of nodes (the size of file is 100MB and the number of data chunks is 1000)

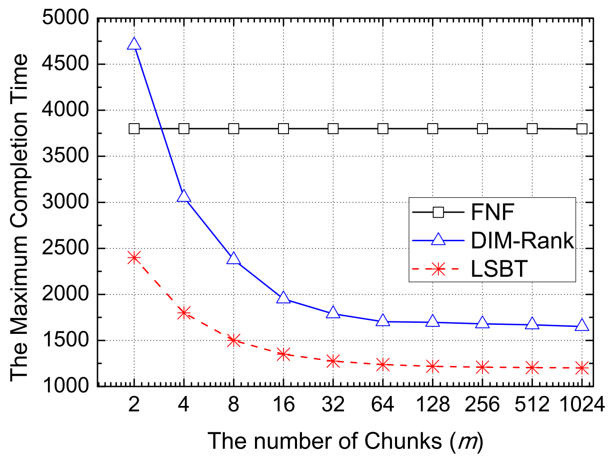


Fig. 9. The number of chunks versus the maximum completion time (the size of file is 100MB and the number of nodes is 100, and the unit on y-axis is second)

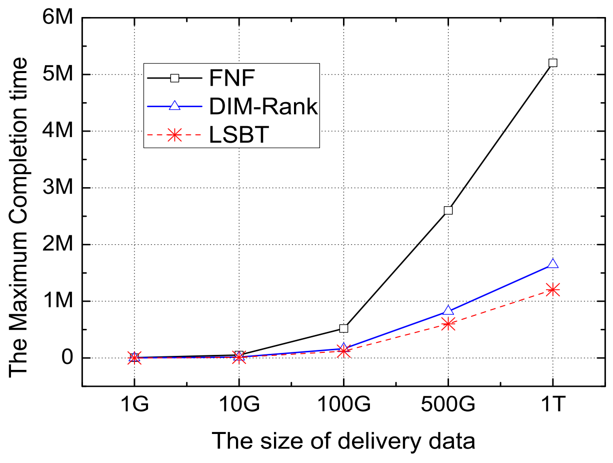


Fig. 11. The maximum completion time versus the size of delivery data (the number of data chunks is 1000 and the number of nodes is 1000, and the unit on y-axis is million milliseconds)

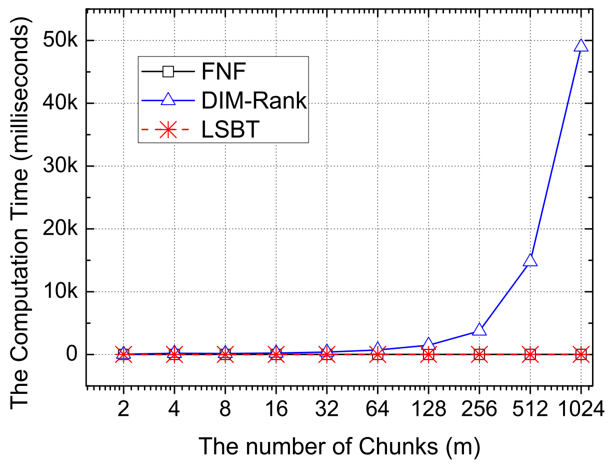


Fig. 10. The number of chunks versus the computation time (the size of file is 100MB and the number of nodes is 100, and the unit on y-axis is thousand milliseconds)

FNF heuristics gives a poor performance, which is expected because FNF does not take the advantage of the pipeline manner. We notice that the computation time of DIM-Rank is significant, and it is because the time complexity of DIM-Rank is O$(m \times nlogn)^2$. In a $n = 10000$ network, the computation time of DIM-Rank requires almost 15 hours (on a sever with Intel Xeon 2.33GHz and 8GB RAM). Thus we do not plot the result of n =100000 network.

Figure 9 plots the effect when the number of data chunks (m) is increased. The size of file is 100MB and the number of nodes is 100. We can see the result of FNF do not depend on the value of m. Note that the x-axis is also a log-scale ($log_2 m$). The maximum completion time of LSBT is significantly lower (at least about 60%) than the one performed by the two other algorithms. Another interesting remark is that DIM-Rank performs worse than FNF when $m = 2$. It is because in the concept of DIM-Rank algorithm, it prefers to let every node obtain a data chunk first. Thus the low-capacity nodes may slow down the maximum completion time.

Figure 10 plots the effect of the computation time when the number of data chunks (m) is increased. The size of file is 100MB and the number of nodes is 100. We can see the results of LSBT and FNF do not depend on the value of m. Note that the x-axis is also a log-scale ($log_2 m$). The computation time of LSBT is significantly lower than the one performed by DIM-Rank algorithms.

Figure 11 plots the effect when the size of the delivery data is increased. The number of data chunks is 1000 and the number of nodes is 1000. The maximum completion time of LSBT is significantly lower than the one performed by the two other algorithms when the size of the delivery data is increased from 1 gigabyte to 1 terabyte. By the simulation results, LSBT shows its ability to offer high performance of big data delivery in heterogeneous datacenter networks. Note that the y-axis of Figure 11 is million seconds.

6 Related work

The data broadcasting problem established by Edmonds [33] since the 1970s and has been studied in many articles. The broadcast problem is the core of every data distribution system, especially in peer-to-peer (P2P) overlay fields, it is of great interest to current efficient P2P data distribution systems, based on a tree or mesh design [21]–[23]. While there is much work on system design and measurement studies of P2P data distribution systems [24], few papers work on theoretical analysis and fundamental limitations of P2P data distribution systems. Ezovski *et al.* [34] proposed an optimal network topology and the associated scheduling policy to achieve the min-min times, by assuming that the file is broken into infinitesimally small chunks such that there is almost no forwarding delay. The authors claimed that the proposed scheme which achieves min-min times can also achieve the minimum average finish time. However, Chang *et al.* [35] disproved the claim in [34]. In [13], the authors propose several distributed algorithms to optimize the throughput of a broadcasting operation. However, they do not consider degree constraints in each node. In [14], Beaumont *et al.* considered the maximizing throughput problem of broadcasting a large message in heterogenous networks. They introduced the bounded degree multi-port model to model the capabilities of the nodes and proved that the data broadcasting problem of maximizing the overall throughput is NP-Complete. Liu *et al.* [36] studied the maximum streaming rate problem of peer-assisted streaming systems. They use a multi-tree formulation and consider per-tree degree bounds. However, they assume that the degrees of all nodes are equal, except for the source node which has unbounded degree. The same authors consider global per-node degree bounds in the article [37].

7 Conclusion

In this paper, we studied the classical data broadcasting problem from an algorithmic point of view. We formalized the problem into the LockStep tree (LSBT) model in which we consider at the same time the design of such a single overlay tree (with a fixed uplink rate) and the maximum completion time of this model. To the best of our knowledge, this work is the first study to investigate the relation between a single overlay tree with a fixed uplink rate and the maximum completion time both in heterogeneous networks. In addition, we envision that our LSBT could be well-suited for a host of applications. We also proposed a novel polynomial-time algorithm to select the optimal uplink rate r^* for building an optimal LSBT. The time complexity of our algorithm is $\text{O}(n \log^2 n)$. Interesting future work involves obtaining good heuristics to the data broadcasting problem. A more challenging version of the problem is to demand multiple LSBTs, we leave it as an interesting future direction. Moreover, in inter-datacenter networks, how to build an optimal LSBT regarding to the physical network topology is another interesting problem.

Acknowledgment

The work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC100-2219-E-001-002 and NSC99-2221-E-001-013-MY3. The authors would like to thank Prof. Kwei-Jay Lin for comments on an earlier draft of this paper. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the manuscript.

References

[1] R. E. Bryant, R. H. Katz, and E. D. Lazowska, “Big-data computing: Creating revolutionary break throughs in commerce, science, and society,” *In Computing Research Initiatives for the 21st Century.*, 2008.

[2] A. Szalay and J. Gray, “2020 computing: Science in an exponential world,” *Nature 440, 413-414, March*, 2006.

[3] G. Brumfiel, “High-energy physics: Down the petabyte highway,” *Nature 469, 282-283 January*, 2011.

[4] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *Proc. of Operating Systems Design and Implementation (OSDI)*, 2004.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, , and R. E. Gruber, “Bigtable: A distributed storage system for structured data,” *Proc. of Operating Systems Design and Implementation (OSDI)*, 2006.

[6] W. D. Hillis and G. L. Steele, Jr., “Data parallel algorithms,” *Communications of the ACM*, vol. 29, pp. 1170–1183, December 1986.

[7] U. Rencuzogullari and S. Dwarkadas, “Dynamic adaptation to available resources for parallel computing in an autonomous network of workstations,” *Proc. of ACM SIGPLAN PPoPP*, 2001.

[8] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, “Managing data transfers in computer clusters with orchestra,” *Proc. of ACM Special Interest Group on Data Communication (SIGCOMM)*, pp. 98–109, 2011.

[9] D. Nukarapu, B. Tang, L. Wang, and S. Lu, “Data replication in data intensive scientific applications with performance guarantee,” *IEEE Transactions on Parallel and Distributed Systems*, aug. 2011.

[10] C. Peng, M. Kim, Z. Zhang, and H. Lei, “Vdn: Virtual machine image distribution network for cloud data centers,” *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[11] S. Khuller and Y.-A. Kim, “Broadcasting in heterogeneous networks,” *Algorithmica*, vol. 48, no. 1, Mar. 2007.

[12] J. Mundinger, R. Weber, and G. Weiss, “Optimal scheduling of peer-to-peer file dissemination,” *Journal of Scheduling*, vol. 11, no. 2, 2008.

[13] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, “P2p streaming capacity under node degree bound,” *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[14] O. Beaumont, L. Eyraud-Dubois, and S. K. Agrawal, “Broadcasting on large scale heterogeneous platforms under the bounded multi-port model,” *Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2011.

[15] S. M. Hedetniemi, S. T. Hedetniemi, and A. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, 1988.

[16] P. Liu, "Broadcast scheduling optimization for heterogeneous cluster systems," *J. Algorithms*, vol. 42, no. 1, Jan. 2002.

[17] K. Wang, J. Li, and L. Pan, "Fast file dissemination in peer-to-peer networks with upstream bandwidth constraint," *Future Generation Computer Systems*, vol. 26, July 2010.

[18] K.-S. Goetzmann, T. Harks, M. Klimm, and K. Miller, "Optimal file distribution in peer-to-peer networks," *Proc. of The 22nd International Symposium on Algorithms and Computation (ISAAC)*, 2011.

[19] M. Deshpande, N. Venkatasubramanian, and S. Mehrotra, "Heuristics for flash-dissemination in heterogenous networks," *Proc. of the 13th international conference on High Performance Computing*, 2006.

[20] B. Cohen, "Incentives build robustness in bittorrent," *Proc. of ACM Workshop on Economics of peer-to-peer systems (P2PECON)*, 2003.

[21] 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 Symposium on Operating Systems Principles (SOSP)*, 2003.

[22] D. Kosti, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," *Proc. of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[23] C.-J. Wu, C.-Y. Li, K.-H. Yang, J.-M. Ho, and M.-S. Chen, "Time-critical data dissemination in cooperative peer-to-peer systems," *Proc. of IEEE Global Telecommunications (GLOBECOM)*, 2009.

[24] A. Passarella, "A survey on content-centric technologies for the current internet: Cdn and p2p solutions," *Computer Communications*, 2012.

[25] M. A. Brown, "Traffic control howto. chapter 6. classless queuing disciplines," *http://tldp.org/HOWTO/Traffic-Control-HOWTO/classless-qdiscs.html*, 2006.

[26] A. Cayley, "A theorem on trees," *Quarterly Journal of Mathematics*, 1889.

[27] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving a bittorrent networks performance mechanisms," *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[28] R. Thommes and M. Coates, "Bittorrent fairness: Analysis and improvements," *Proc. of Workshop on the Internet, Telecommunications and Signal Processing (WITSP)*, December 2005.

[29] Murder, "https://github.com/lg/murder."

[30] S. ul Islam, K. Stamos, J.-M. Pierson, and A. Vakali, "Utilization-aware redirection policy in cdn: A case for energy conservation," *Proc. of Information and Communication on Technology for the Fight against Global Warming*, 2011.

[31] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," *Proc. of Multimedia Computing and Networking (MMCN)*, 2002.

[32] M. Banikazemi, V. Moorthy, and D. Panda, "Efficient collective communication on heterogeneous networks of workstations," *Proc. ofIEEE International Conference on Parallel Processing*, pp. 460–467, 1998.

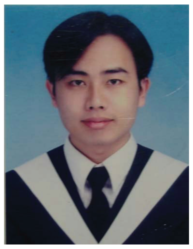
[33] J. Edmonds, "Edge-disjoint branchings, in combinatorial algorithms," *Algorithmics Press*, 1972.

[34] G. M. Ezovski, A. Tang, and L. L. H. Andrew, "Minimizing average finish time in p2p networks," *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

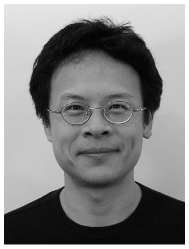
[35] C. Chang, T. Ho, M. Effros, M. Medard, and B. Leong, "Issues in peer-to-peer networking: a coding optimization approach," *Proc. of IEEE International Symposium on Network Coding (NetCod)*, 2010.

[36] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," *Proc. of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2008.

[37] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou, "P2p streaming capacity under node degree bound," *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2010.



Chi-Jen Wu works as a Postdoc researcher at the Institute of Information Science of Academia Sinica in Taiwan. He received his Ph.D. degree in EECS from National Taiwan University in 2012. Chi-Jen was a research assistant at the Institute of Information Science of Academia Sinica before completing his PhD. His research interests include Content-Centric Networking, Future Internet, Peer-to-Peer systems and Mobile Data Management. He is a student member of the ACM.



Chin-Fu Ku received the B.S. degree in Computer Science from National Chiao Tung University, Taiwan, R.O.C., in 1994, and the M.S. degree in Computer Science and Engineering from Yuan Ze University, Taiwan, R.O.C., in 1996. He is a Postdoc researcher at the Institute of Information Science of Academia Sinica in Taiwan. His research interests include: Internet protocol, multimedia streaming and performance modeling.

Jan-Ming Ho received his Ph.D. degree in electrical engineering and computer science from Northwestern University in 1989. He received his B.S. in electrical engineering from National Cheng Kung University in 1978 and his M.S. at Institute of electronics of National Chiao Tung University in 1980. Dr. Ho joined the Institute of Information Science, Academia Sinica as associate research fellow in 1989, and was promoted to research fellow in 1994. He visited IBM's T. J. Watson Research Center in summer 1987 and summer 1988, and the Leonardo Fibonacci Institute for the Foundations of Computer Science, Italy in summer 1992. In 2004-2006, he was jointly appointed by National Science Council, Taiwan, where he served as Director General of Division of Planning and Evaluation. His research interests cover 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. He is Associate Editor of IEEE Transaction on Multimedia. He was Program Chair of Symposium on Real-time Media Systems, Taipei, 1994 - 1998, General Co-Chair of International Symposium on Multi-Technology Information Processing, 1997 and will be General Co-Chair of IEEE RTAS 2001. He was also steering committee member of VLSI Design/CAD Symposium, and program committee member of several previous conferences including ICDCS 1999, and IEEE Workshop on Dependable and Real-Time E-Commerce Systems (DARE'98), etc. In domestic activities, he is Program Chair of Digital Archive Task Force Conference, the First Workshop on Digital Archive Technology, Steering Committee Member of the 12th VLSI Design/CAD Symposium and International Conference on Open Source 2001, and is also Program Committee Member of the 13th Workshop on Object-Oriented Technology and Applications, the 8th Workshop on Mobile Computing, 2001 Summer Institute on Bioinformatics, and Workshop on Information Society and Digital Divide.

Ming-Syan Chen received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, and the M.S. and Ph.D. degrees in Computer, Information and Control Engineering from The University of Michigan, Ann Arbor, MI, USA, in 1985 and 1988, respectively. He is now a Distinguished Research Fellow and the Director of Research Center of Information Technology Innovation (CITI) in the Academia Sinica, Taiwan, and is also a Distinguished Professor jointly appointed by EE Department, CSIE Department, and Graduate Institute of Communication Eng. (GICE) at National Taiwan University. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA from 1988 to 1996, the Director of GICE from 2003 to 2006, and also the President/CEO of Institute for Information Industry (III), which is one of the largest organizations for information technology in Taiwan, from 2007 to 2008. His research interests include databases, data mining, mobile computing systems, and multimedia networking, and he has published more than 270 papers in his research areas. In addition to serving as program chairs/vice-chairs and keynote/tutorial speakers in many international conferences, Dr. Chen was an associate editor of IEEE TKDE and also JISE, is currently on the editorial board of Very Large Data Base (VLDB) Journal, Knowledge and Information Systems (KAIS) Journal, and International Journal of Electrical Engineering (IJEE), and is a Distinguished Visitor of IEEE Computer Society for Asia-Pacific from 1998 to 2000, and also from 2005 to 2007. He holds, or has applied for, eighteen U.S. patents and seven ROC patents in his research areas. He is a recipient of the NSC (National Science Council) Distinguished Research Award, Pan Wen Yuan Distinguished Research Award, Teco Award, Honorary Medal of Information, and K.-T. Li Research Breakthrough Award for his research work, and also the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He also received numerous awards for his research, teaching, inventions and patent applications. Dr. Chen is a Fellow of ACM and a Fellow of IEEE.