On Generalized Gossiping and Broadcasting*

Samir Khuller, Yoo-Ah Kim, and Yung-Chun (Justin) Wan**

Department of Computer Science and Institute for Advanced Computer Studies,
University of Maryland, College Park, MD 20742.
{samir,ykim,ycwan}@cs.umd.edu

Abstract. The problems of gossiping and broadcasting have been widely studied. The basic gossip problem is defined as follows: there are n individuals, with each individual having an item of gossip. The goal is to communicate each item of gossip to every other individual. Communication typically proceeds in rounds, with the objective of minimizing the number of rounds. One popular model, called the telephone call model, allows for communication to take place on any chosen matching between the individuals in each round. Each individual may send (receive) a single item of gossip in a round to (from) another individual. In the broadcasting problem, one individual wishes to broadcast an item of gossip to everyone else. In this paper, we study generalizations of gossiping and broadcasting. The basic extensions are: (a) each item of gossip needs to be broadcast to a specified subset of individuals and (b) several items of gossip may be known to a single individual. We study several problems in this framework that generalize gossiping and broadcasting. Our study of these generalizations was motivated by the problem of managing data on storage devices, typically a set of parallel disks. For initial data distribution, or for creating an initial data layout we may need to distribute data from a single server or from a collection of sources.

1 Introduction

The problems of Gossiping and Broadcasting have been the subject of extensive study [21, 15, 17, 3, 4, 18]. These play an important role in the design of communication protocols in various kinds of networks. The *gossip problem* is defined as follows: there are n individuals. Each individual has an item of gossip that they wish to communicate to everyone else. Communication is typically done in rounds, where in each round an individual may communicate with at most one other individual (also called the telephone model). There are different models that allow for the full exchange of all items of gossip known to each individual in a single round, or allow the sending of only one item of gossip from one to the other (half-duplex) or allow each individual to send an item to the individual they are communicating with in this round (full-duplex). In addition, there may be a communication graph whose edges indicate which pairs of individuals are allowed to communicate in each round. (In the classic gossip problem, communication may take place between any pair of individuals; in other words, the communication graph is the complete graph.) In the *broadcast problem*, one individual needs to convey an item of gossip to every other individual. The two parameters typically used to evaluate the algorithms for this problem are: the number of communication rounds, and the total number of telephone calls placed.

The problems we study are generalizations of the above mentioned gossiping and broadcasting problems. The basic generalizations we are interested in are of two kinds (a) each item of gossip needs to be communicated to only a subset of individuals, and (b) several items of gossip may be known to one individual. Similar generalizations have been considered before [23, 25]. (In Section

* This research was supported by NSF Awards CCR-9820965 and CCR-0113192.
** FAX: (301) 314-1353

1.2 we discuss in more detail the relationships between our problem and the ones considered in those papers.)

There are four basic problems that we are interested in. Before we define the problems formally, we discuss their applications to the problem of creating data layouts in parallel disk systems. The communication model we use is the half-duplex telephone model, where only one item of gossip may be communicated between two communicating individuals during a single round. Each individual may communicate (either send or receive an item of data) with at most one other individual in a round. This model best captures the connection of parallel storage devices that are connected on a network and is most appropriate for our application.

We now briefly discuss applications for these problems, as well as prior related work on data migration. To deal with high demand, data is usually stored on a parallel disk system. Data objects are often replicated within the disk system, both for fault tolerance as well as to cope with demand for popular data [29, 5]. Disks typically have constraints on storage as well as the number of clients that can simultaneously access data from it. Approximation algorithms have been developed [26, 27, 12, 19] to map known demand for data to a specific data layout pattern to maximize utilization[1]. In the layout, we not only compute how many copies of each item we need, but also a layout pattern that *specifies the precise subset of items on each disk.* The problem is NP-hard, but there is a polynomial time approximation scheme [12]. Hence given the relative demand for data, the algorithm computes an almost optimal layout. For example, we may wish to create this layout by copying data from a single source that has all the data initially. Or the data may be stored at different locations initially—these considerations lead to the different problems that we consider.

In our situation, each individual models a **disk** in the system. Each item of gossip is a **data item** that needs to be transferred to a set of disks. If each disk had exactly one data item, and needs to copy this data item to every other disk, then it is exactly the problem of gossiping.

Different communication models can be considered based on how the disks are connected. We use the same model as in the work by [13, 1] where the disks may communicate on any matching; in other words, the underlying communication graph is complete. For example, *Storage Area Networks* support a communication pattern that allows for devices to communicate on a specified matching.

Suppose we have N disks and Δ data items. The problems we are interested in are:

1. **Single-source broadcast**. There are Δ data items stored on a single disk (the source). We need to broadcast all items to all $N-1$ remaining disks.
2. **Single-source multicast**. There are Δ data items stored on a single disk (the source). We need to send data item i to a specified subset D_i of disks. Figure 1 gives an example when Δ is 4.
3. **Multi-source broadcast**. There are Δ data items, each stored separately at a single disk. These need to be broadcast to all disks. We assume that data item i is stored on disk i, for $i = 1 \ldots \Delta$.
4. **Multi-source multicast**. There are Δ data items, each stored separately at a single disk. Data item i needs to be sent to a specified subset D_i of disks. We assume that data item i is stored on disk i, for $i = 1 \ldots \Delta$.

We do not discuss the first problem in any detail since this was solved by [8, 10]. For the multi-source problems, there is a sub-case of interest, namely when the source disks are not in any subset D_i. For this case we can develop better bounds.

[1] Utilization refers to the total number of clients that can be assigned to a disk that contains the data they want.

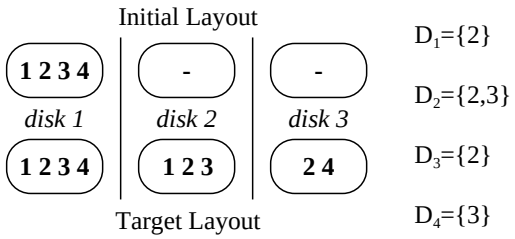

Fig. 1. An initial and target layouts, and their corresponding D_i's of a single-source multicast instance.

1.1 Contributions

In Section 2 we define the basic model of communication and the notation used in the paper. Let N be the number of disks and Δ be the number of items. The main results that we show in this paper are:

Theorem 1.1. *For the single-source multicast problem we design a polynomial time algorithm that outputs a solution where the number of rounds is at most $OPT + \Delta$.*

Theorem 1.2. *For the multi-source broadcast problem we design a polynomial time algorithm that outputs a solution where the number of rounds is at most $OPT + 3$.*

Theorem 1.3. *For the multi-source multicast problem we design a polynomial time algorithm that outputs a solution where the number of rounds is at most $4OPT + 2$. Moreover, we show that this problem is NP-hard.*

Theorem 1.4. *For the multi-source multicast problem we also design a polynomial time algorithm that outputs a solution where the number of rounds is at most $(3 + o(1))OPT$.*

For all the above algorithms, we move data only to disks that need the data. Thus we use no bypass (intermediate) nodes as holding points for the data. If bypass nodes are allowed, we have the following result:

Theorem 1.5. *For the multi-source multicast problem allowing bypass nodes we design a polynomial time algorithm that outputs a solution where the number of rounds is at most $3OPT + 6$.*

1.2 Related Work

One general problem of interest is the **data migration problem** when data item i resides in a specified (source) subset S_i of disks, and needs to be moved to a (destination) subset D_i. This problem is more general than the Multi-source multicast problem where we assumed that $|S_i| = 1$ and that all the S_i's are disjoint. For the data migration problem we have developed a 9.5-approximation algorithm [20]. While this problem is a generalization of all the problems we study in this paper (and clearly also NP-hard since even the special case of multi-source multicast is NP-hard), the bounds in [20] are not as good. The methods used for single-source multicast and multi-source broadcast are completely different from the algorithm in [20]. Using the methods in [20] one cannot obtain additive bounds from the optimal solution. The algorithm for multi-source multicast presented here is a simplification of the algorithm developed in [20], and we also obtain a much better approximation factor of 4. In addition, by allowing bypass nodes we can improve the bounds further.

Many generalizations of gossiping and broadcasting have been studied before. For example, the paper by Liben-Nowell [23] considers a problem very similar to multi-source multicast with $\Delta = N$. However, the model that he uses is different than the one that we use. In his model, in each telephone call, a pair of users can exchange all the items of gossip that they know. The objective is to simply minimize the total number of phone calls required to convey item i of gossip to set D_i of users. In our case, since each item of gossip is a data item that might take considerable time to transfer between two disks, we cannot assume that an arbitrary number of data items can be exchanged in a single round. Several other papers use the same telephone call model [2, 7, 14, 18, 30]. Liben-Nowell [23] gives an exponential time exact algorithm for the problem.

Other related problems that have been studied are the set-to-set gossiping problem [22, 25] where we are given two possibly intersecting sets A and B of gossipers and the goal is to minimize the number of calls required to inform all gossipers in A of all the gossip known to members in B. The work by [22] considers minimizing both the number of rounds as well as the total number of calls placed. The main difference is that in a single round, an arbitrary number of items may be exchanged. For a complete communication graph they provide an exact algorithm for the minimum number of calls required. For a tree communication graph they minimize the number of calls or number of rounds required. Liben-Nowell [23] generalizes this work by defining for each gossiper i the set of relevant gossip that they need to learn. This is just like our multi-source multicast problem with $\Delta = N$, except that the communication model is different, as well as the objective function. The work by [9] also studies a set to set broadcast type problem, but the cost is measured as the total cost of the broadcast trees (each edge has a cost). The goal is not to minimize the number of rounds, but the total cost of the broadcast trees. In [11] they also define a problem called scattering which involves one node broadcasting distinct messages to all the other nodes (very much like our single source multicast, where the multicast groups all have size one and are disjoint).

As mentioned earlier, the single source broadcast problem using the same communication model as in our paper was solved by [8, 10].

2 Models and Definitions

We have N disks and Δ data items. Note that after a disk receives item i, it can be a source of item i for other disks that have not received the item as yet. Our goal is to find a schedule using the minimum number of rounds, that is, to minimize the total amount of time to finish the schedule. We assume that the underlying network is connected and the data items are all the same size, in other words, it takes the same amount of time to migrate an item from one disk to another. The crucial constraint is that each disk can participate in the transfer of only one item—either as a sender or receiver. Moreover, as we do not use any bypass nodes, all data is only sent to disks that desire it.

Our algorithms make use of a known result on edge coloring of multi-graphs. Given a graph G with max degree Δ_G and multiplicity μ the following result is known (see [6] for example). Let χ' be the edge chromatic number of G.

Theorem 2.1. *(Vizing [31]) If G has no self-loops then $\chi' \leq \Delta_G + \mu$.*

3 Single-Source Multicasting

In this section, we consider the case where there is one source disk s that has all Δ items and others do not have any item in the beginning. For the case of *broadcasting* all items, it is known that there is a schedule which needs $2\Delta - 1 + \lfloor \log N \rfloor$ rounds for odd N and $\lceil \frac{\Delta(N-1) - 2^{\lfloor \log_2 N \rfloor} + 1}{\lfloor N/2 \rfloor} \rceil + \lfloor \log N \rfloor$

rounds for even N [8, 10] and this is optimal. We develop an algorithm that can be applied when D_i is an arbitrary subset of disks. The number of rounds required by our algorithm is at most $\Delta + OPT$ where OPT is the minimum number of rounds required for this problem. Our algorithm is obviously a 2-approximation for the problem, since Δ is a lower bound on the number of rounds required by the optimal solution.

3.1 Outline of the Algorithm

Without loss of generality, we assume that $|D_1| \geq |D_2| \geq \cdots \geq |D_\Delta|$ (otherwise renumber the items). Let $|D_i| = 2^{d_i^1} + 2^{d_i^2} + \cdots + 2^{d_i^{m_i}}$ where $d_i^j (j = 1, 2, \ldots, m_i)$ are integers and $d_i^j > d_i^{j+1}$. (In other words, we consider the bit representation of each $|D_i|$ value.)

Our algorithm consists of two phases.

Phase I. In the first phase, we want to make exactly $\lfloor |D_i|/2 \rfloor$ copies for all items i. At the t-th round, we do the following:

1. If $t \leq \Delta$, copy item t from source s to a disk in D_t.
2. For items j ($j < t$), double the number of copies unless the number of copies reaches $\lfloor |D_j|/2 \rfloor$. In other words, every disk having an item j makes another copy of it if the number of copies of item j is no greater than $2^{d_j^1 - 2}$, and when it becomes $2^{d_j^1 - 1}$, then only $\lfloor |D_j|/2 \rfloor - 2^{d_j^1 - 1}$ disks make copies, and thus the number of copies of item i becomes $\lfloor |D_i|/2 \rfloor$.

Phase II. At t-th round, we finish the migration of item t. Each item j has $\lfloor |D_j|/2 \rfloor$ copies. We finish migrating item t by copying from the current copies to the remaining $\lfloor |D_t|/2 \rfloor$ disks in D_t which did not receive item t as yet, and we use the source disk if $|D_t|$ is odd.

Figure 2 and Figure 3 show an example of data transfers taken in Phase 1 and Phase 2, respectively, where $|D_1|$, $|D_2|$ and $|D_3|$ are 16, 12 and 8, respectively. It is easy to see that Phase II can be scheduled without conflicts because we deal with only one item each round. But in Phase I, migration of several items happen at the same time and D_i's can overlap. Therefore, we may not be able to satisfy the requirement of each round if we arbitrarily choose the disks to receive items. We show that we can finish Phase I successfully without conflicts by choosing disks carefully.

3.2 Details of Phase I

Let D_i^p be the disks in D_i that participate in either sending or receiving item i at the $(i+p)$-th round. D_i^0 is the first disk receiving i from the source s and

$$|D_i^p| = \begin{cases} 2^p & \text{if } p \leq d_i^1 - 1 \\ 2\lfloor \frac{|D_i|}{2} \rfloor - 2^{d_i^1} & \text{if } p = d_i^1 \end{cases}$$

At $(i+p)$-th round, disks in $D_j^{i+p-j} (i + p - d_j^1 \leq j \leq \min(i+p, \Delta))$ either send or receive item j at the same time. To avoid conflicts, we decide which disks belong to D_i^p before starting migration. If we choose disks from $D_i \bigcap D_j$ for D_i^p $(j > i)$, it may interfere with the migration of D_j. Therefore, when we build D_i^p, we consider $D_j^{p'}$ where $j > i$ and $p' \leq p$. Also note that since each disk receiving an item should have its corresponding sender; half of D_i^p should have item i as senders and another half should not have item i as receivers.

We build D_Δ^p first. Choose $2\lfloor |D_\Delta|/2 \rfloor - 2^{d_\Delta^1}$ disks for $D_\Delta^{d_\Delta^1}$ and $2^{d_\Delta^1 - 1}$ disks for $D_\Delta^{d_\Delta^1 - 1}$ from D_Δ. When we choose $D_\Delta^{d_\Delta^1 - 1}$, we should include the half of $D_\Delta^{d_\Delta^1}$ (that will be senders at $(\Delta + d_\Delta^1)$-th round) and exclude the remaining half of $D_\Delta^{d_\Delta^1}$ (that will be receivers at $(\Delta + d_\Delta^1)$-th round). And then build $D_\Delta^p (p < d_\Delta^1 - 1)$ by taking any subset of D_Δ^{p+1}.

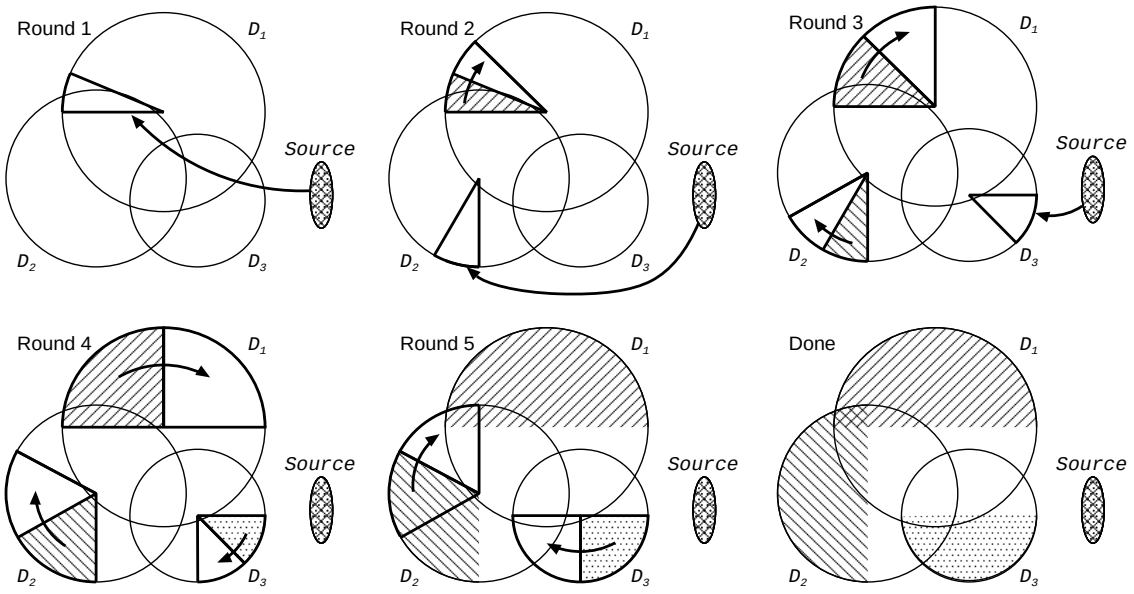

Fig. 2. An example of Phase I when all $|D_i|$ are even

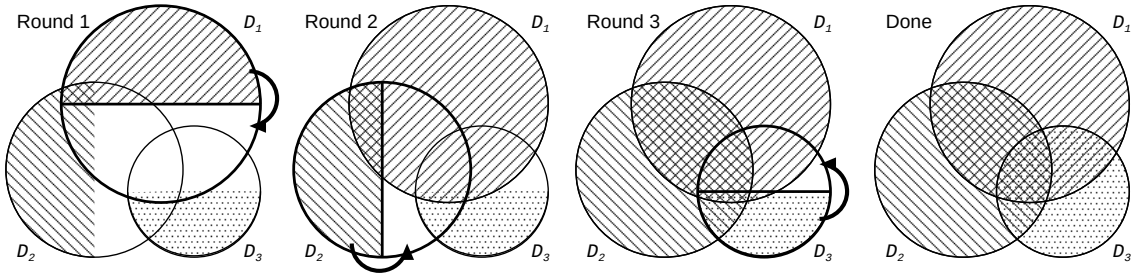

Fig. 3. An example of Phase II when all $|D_i|$ are even

Now given $D_j^{p'}$ $(i < j \leq \Delta)$, we decide D_i^p as follows: Define D_i' to be disks in D_i which do not have any item $j(> i)$ after $(i + d_i^1)$-th round. In the same way, define D_i'' to be disks in D_i which do not have any item $j(> i)$ after $(i + d_i^1 - 1)$-th round. Formally, since all disks in $\bigcup_{p=0}^{p'} D_j^p$ have item j after $(j + p')$-th rounds, $D_i' = D_i - \bigcup_{j=i+1}^{\Delta}(\bigcup_{p=0}^{i+d_i^1-j} D_j^p)$ and $D_i'' = D_i - \bigcup_{j=i+1}^{\Delta}(\bigcup_{p=0}^{i+d_i^1-1-j} D_j^p)$. As shown in Figure 4, we choose $D_i^{d_i^1}$ from D_i' and also $D_i^{d_i^1-1}$ from D_i'', by which we can avoid conflicts. Also, half of $D_i^{d_i^1}$ should be included in $D_i^{d_i^1-1}$ (to be senders) and the remaining half should be excluded from $D_i^{d_i^1-1}$ (to be receivers). We make $D_i^p (p < d_i^1 - 1)$ by choosing any subset of disks from D_i^{p+1}.

Lemma 3.1. *We can find a migration schedule by which we perform every round in phase I without conflicts.*

Proof. First we show that there are enough disks to build D_i^p as described above. Because $|\bigcup_{p=0}^{p'} D_j^p| \leq 2^{p'}$,

$$\begin{aligned}|D_i''| &= |D_i - \bigcup_{j=i+1}^{\Delta} (\bigcup_{p=0}^{i+d_i^1-1-j} D_j^p)| \\ &\geq |D_i| - \sum_{j=i+1}^{\Delta} 2^{i+d_i^1-1-j} \\ &\geq |D_i| - \sum_{m=0}^{d_i^1-2} 2^m > |D_i| - 2^{d_i^1-1}\end{aligned}$$

Therefore, even after excluding $\lfloor |D_i|/2 \rfloor - 2^{d_i^1-1}$ disks in D_i' from D_i'', we have at least $|D_i|/2 = 2^{d_i^1-1} + 2^{d_i^2-1} + \cdots + 2^{d_i^{m_i}-1}$ disks, from which we can take $2^{d_i^1-1}$ disks for $D_i^{d_i^1}$. Also we know that

$$|D_i'| = |D_i - \bigcup_{j=i+1}^{\Delta} (\bigcup_{p=0}^{i+d_i^1-j} D_j^p)| > |D_i| - 2^{d_i^1}.$$

Because we only need $2\lfloor |D_i|/2 \rfloor - 2^{d_i^1}$ disks for $D_i^{d_i^1-1}$, we have enough disks to choose.

Now we argue that there is no conflict in performing migration if we do migration according to D_i^p. Since $D_i^{d_i^1} \subset D_i'$ and $D_i' \bigcap D_j^{i+d_i^1-j} = \emptyset$ $(j > i)$, there is no conflict between i and j at $(i+d_i^1)$-th round. For $p \leq d_i^1 - 1$, since $D_i^p \subset D_i''$ and $D_i'' \bigcap D_j^{i+p-j} = \emptyset$ $(j > i)$, there is no conflict between i and j at $(i+p)$-th round. Therefore, we can perform migration in Phase I without conflicts. □

3.3 Analysis

We prove that our algorithm uses at most Δ more rounds than the optimal solution for single-source multicasting. Let us denote the optimal makespan of an migration instance I as $C(I)$.

Theorem 3.2. *For any migration instance I, $C(I) \geq \max_{1 \leq i \leq \Delta}(i + \lfloor \log |D_i| \rfloor)$.*

Proof. Consider the instance where there is no overlap among D_i's. After a disk in D_i receives i from s for the first time, we need at least $\lfloor \log |D_i| \rfloor$ more rounds to make all disks in D_i receive i even if s copies item i several times after the first copy. Therefore, $C(I) \geq \max_{1 \leq i \leq \Delta}(f(i) + \lfloor \log |D_i| \rfloor)$ where $f(i)$ is the round when D_i receives the first copy from s. Because s can be involved in copying only one item at a time, $f(i) \neq f(j)$ if $i \neq j$. Also copying the same item from s more than once during the first Δ rounds will only increase $f(i)$ of some sets. Therefore, $f(i)$ should be a permutation of $1, \ldots, \Delta$ to minimize the value. Now we show that $\max_{1 \leq i \leq \Delta}(f(i) + \lfloor \log |D_i| \rfloor) \geq \max_{1 \leq i \leq \Delta}(i + \lfloor \log |D_i| \rfloor)$ for any permutation $f(i)$. Suppose there is a set D_i that $f(i) \neq i$ when $\max_{1 \leq i \leq \Delta}(f(i) + \lfloor \log |D_i| \rfloor)$ is minimum. Let D_i be the set which have the smallest $f(i)$ among such sets. Then $f(i) < i$ and there should be a D_j such that $j = f(i)$ and $f(j) > j$. Even if we exchange the order of two sets, the value does not increase because

$$\begin{aligned}\max(f(i) + \lfloor \log |D_i| \rfloor, f(j) + \lfloor \log |D_j| \rfloor) &= f(j) + \lfloor \log |D_j| \rfloor \\ &\geq \max(j + \lfloor \log |D_j| \rfloor, f(j) + \lfloor \log |D_i| \rfloor).\end{aligned}$$

Thus when $f(i) = i$ for all i, $\max_{1 \leq i \leq \Delta}(f(i) + \lfloor \log |D_i| \rfloor)$ is minimized. □

Lemma 3.3. *The total makespan of our algorithm is at most* $\max_{1 \le i \le \Delta}(i + \lfloor \log |D_i| \rfloor) + \Delta$.

Proof. In the phase I, D_i receives i from s at i-th round for the first time. Because the number of copies is doubled after then until it reaches $\lfloor |D_i|/2 \rfloor$, the number of copies of item i reaches $\lfloor |D_i|/2 \rfloor$ in $i + \lfloor \log |D_i| \rfloor$ rounds. Phase II takes at most Δ rounds because we finish one item at a round. Therefore, the lemma follows. □

Corollary 3.4. *The total makespan of our algorithm is at most the optimal makespan plus* Δ.

Proof. Follows from Lemma 3.2 and Lemma 3.3. □

Theorem 3.5. *We have a 2-approximation algorithm for the single-source multicasting problem.*

Proof. Because $\Delta \le \max_{1 \le i \le \Delta}(i + \lfloor \log |D_i| \rfloor)$, the algorithm is 2-approximation. □

4 Multi-Source Broadcasting

We assume that we have N disks. Disk i, $1 \le i \le \Delta$, has an item numbered i. The goal is to send each item i to all N disks, for all i. We present an algorithm which performs no more than 3 extra rounds than the optimal solution.

4.1 Algorithm Multi-Source Broadcast

1. We divide N disks into Δ disjoint sets G_i such that disk $i \in G_i$, for all $i = 1 \ldots \Delta$. Let q be $\lfloor \frac{N}{\Delta} \rfloor$ and r be $N - q\Delta$. $|G_i| = q + 1$ for $i = 1 \ldots r$, and $|G_i| = q$ for $i = r + 1 \ldots \Delta$. Every disk in G_i can receive item i using $\lceil \log |G_i| \rceil$ rounds by doubling the items in each round.
2. We divide all N disks into $q - 1$ groups of size Δ by picking one disk from each G_i, and one group of size $\Delta + r$ which consists of all remaining disks.
3. Consider the first $q - 1$ gossiping groups; each group consists of Δ disks, with each having a distinct item. Using the gossiping algorithm in [4], every disk in the first $q - 1$ groups can receive all Δ items in 2Δ rounds[2].
4. Consider the last gossiping group, there are exactly two disks having items $1, \ldots, r$, while there is exactly one disk having item $r + 1, \ldots, \Delta$. If r is zero, we can finishes all transfers in 2Δ rounds using algorithm in [4]. For non-zero r, we claim that all disks in this gossiping group can receive all items in 2Δ rounds.
   We divide the disks in this gossiping group into 2 groups, G_X and G_Y of size $\Delta - \lfloor \frac{\Delta - r}{2} \rfloor$ and $r + \lfloor \frac{\Delta - r}{2} \rfloor$ respectively. Note that $|G_Y| + 1 \ge |G_X| \ge |G_Y|$. Exactly one disk having items $1, \ldots, r$ appear in each group, disks having item $r + 1, \ldots, \Delta - \lfloor \frac{\Delta - r}{2} \rfloor$ appear in G_X, and the remaining disks (having items $\Delta - \lfloor \frac{\Delta - r}{2} \rfloor + 1, \ldots, \Delta$) appear in G_Y. Note that the size of the two groups differ by at most 1. The general idea of the algorithm is as follows (The details of these step are non-trivial and covered in the proof of Lemma 4.1):
   (a) Algorithm in [4] is applied to each group in parallel. After this step, each disk has all items belong to its group.
   (b) In each round, disks in G_Y send item i to disks in G_X, where i is $\Delta - \lfloor \frac{\Delta - r}{2} \rfloor + 1, \ldots, \Delta$. Note that only disks in G_Y have these items, but not the disks in G_X. Since the group sizes differ by at most 1, the number of rounds required is about the same as the number of items transferred.
   (c) The step is similar to the above step but in different direction. Item i, where i is $r + 1, \ldots, \Delta - \lfloor \frac{\Delta - r}{2} \rfloor$, are copied to G_Y.

   Thus, our algorithm takes $\lceil \log \frac{N}{\Delta} \rceil + 2\Delta$ rounds.

[2] The number of rounds required is 2Δ if Δ is odd, otherwise it is $2(\Delta - 1)$

4.2 Analysis

Lemma 4.1. *For a group of disks of size $\Delta + r$, where $1 \leq r < \Delta$, if every disk has one item, exactly 2 disks have item $1, \ldots r$, and exactly 1 disk has item $r+1, \ldots, \Delta$, all disks can receive all Δ items in 2Δ rounds.*

Proof. We have three cases.
Case I: If $\Delta + r$ is even: No matter $|G_X|$ and $|G_Y|$ is odd or not, Step 4a can be done in $2(\Delta - \frac{\Delta - r}{2})$ rounds because $\Delta - \frac{\Delta - r}{2}$ is the group size. In Step 4b and 4c, we can finish one item in one round since the size of the two groups is the same. All disks can participate in transferring data without any conflict. There are $(\frac{\Delta - r}{2}) + (\Delta - r - \frac{\Delta - r}{2})$ items to be sent in these 2 steps. Thus, the total rounds needed is $(2(\Delta - \frac{\Delta - r}{2})) + (\frac{\Delta - r}{2}) + (\Delta - r - \frac{\Delta - r}{2}) = 2\Delta$.
Case II: If $\Delta + r$ is odd and $|G_X| = \Delta - \frac{\Delta - r - 1}{2}$ is even: Step 4a can be done in $2(\Delta - \frac{\Delta - r - 1}{2} - 1)$ rounds. In Step 4b, $\frac{\Delta - r - 1}{2}$ has to be copied to G_X but $|G_Y|$ is smaller than $|G_X|$ by one. Instead of keeping one disk idle all the time, we shift the disk not receiving any item in each round. After this step finishes, only $\frac{\Delta - r - 1}{2}$ disks in G_X miss 1 item, while other disks in G_X receive all $\frac{\Delta - r - 1}{2}$ items. Using one more round, all disks in G_X can receive all items needed from G_Y. In Step 4c, $\Delta - r - \frac{\Delta - r - 1}{2}$ items have to be copied to G_Y, and we have enough source disks in G_X. Thus, it requires $(2(\Delta - \frac{\Delta - r - 1}{2} - 1)) + (\frac{\Delta - r - 1}{2} + 1) + (\Delta - r - \frac{\Delta - r - 1}{2}) = 2\Delta$ rounds.
Case III: If $\Delta + r$ is odd and $|G_X| = \Delta - \frac{\Delta - r - 1}{2}$ is odd: Since $|G_X|$ is odd, Step 4a takes $2(\Delta - \frac{\Delta - r - 1}{2})$ rounds. We claim that in this step, in addition to receiving items from its group, all disks in G_X, except the disk has item 1 originally, have item Δ, and all disks in G_Y have item $\Delta - \frac{\Delta - r - 1}{2}$ (i.e., the largest numbered item in G_X). We use the algorithm in [4] to form a schedule for G_X with the constraint that (i) the disk has item 1 originally should be idle at the first two rounds, and (ii) the disk which received item $\Delta - \frac{\Delta - r - 1}{2}$, except the disk having item 1 originally, should be idle in the next two rounds. It is not difficult to check that such a schedule exists, and this enforces the disk has item $\Delta - \frac{\Delta - r - 1}{2}$ originally would be idle at the last 2 rounds. We sort disks in G_X according to the item number it has, and label the disks as disk $1, 2, \ldots, \Delta - \frac{\Delta - r - 1}{2}$. We also sort disks in G_Y, but label the disks as $2, 3, \ldots, \Delta - \frac{\Delta - r - 1}{2}$. Disk 1 in G_Y is an imaginary disk which does not exist. Whenever disk x and y in G_X exchange data in the gossiping schedule of G_X, disk x and y in G_Y also exchange data in the same round. Moreover, starting at round 3, the idle disk in G_X, which should have item $\Delta - \frac{\Delta - r - 1}{2}$, will exchange data with the idle disk in G_Y, which should have item Δ. If a disk in G_Y is supposed to exchange data with disk 1 in G_Y (i.e., the imaginary disk), the disk would actually be idle in that round. An example can be found in Figure 5. Note that we just exploit the idle cycles in the gossiping schedule. The number of rounds required is still $2(\Delta - \frac{\Delta - r - 1}{2})$. One disk in G_X always exchanges data with one disk in G_Y except in the first 2 rounds. All disks in G_X and G_Y, except disk 1 in G_X, receive one extra item from other group.

In Step 4b and 4c, the analysis is similar to that in **Case II** except that we save one round in each step because each disk has already received one item from another group in Step 4a. The disk in G_X, which does not have item Δ, can receive it in the last round of Step 4b because $\frac{\Delta - r - 1}{2} + 1 \leq |G_Y|$.

Thus, the total number of rounds is $(2(\Delta - \frac{\Delta - r - 1}{2})) + (\frac{\Delta - r - 1}{2}) + (\Delta - r - \frac{\Delta - r - 1}{2} - 1) = 2\Delta$. □

To show our algorithm is close to optimal, we will show a lower bound of any algorithm for the problem.

Theorem 4.2. *The makespan time of any migration instance of multi-source broadcasting is at least $\lfloor \log \frac{N}{\Delta} \rfloor + 2(\Delta - 1)$.*

Proof. Consider a transfer graph of the optimal solution, where vertices are disks and edge i to j represents one item is copied from disk i to disk j at certain time. For each of the Δ source disks, it needs $\Delta - 1$ items. For each of the remaining $N - \Delta$ disks, it needs all Δ items. Therefore, there should be $\Delta(\Delta - 1) + (N - \Delta)\Delta = \Delta(N - 1)$ edges.

In the initial $\lfloor \log \frac{N}{\Delta} \rfloor$ rounds, some disks have to be idle because of the limited number of sources. For example, if there are x non-empty disks at a certain round, one can perform at most x transfers. If all the transfers send data to other empty disks, one can perform $2x$ transfers in the next round, while other schemes cannot support $2x$ transfers in the next round. Therefore, the best scheme is to keep on doubling all items in each round until all disks have at least one item. This takes $\lfloor \log \frac{N}{\Delta} \rfloor$ rounds. Now, at most $N - \Delta$ transfers are done.

Total degree of the transfer graph after removing the edges corresponding to the first $\lfloor \log \frac{N}{\Delta} \rfloor$ rounds is at least $2(\Delta(N - 1) - (N - \Delta)) = 2N(\Delta - 1)$. Note that each disk can receive or send only 1 item in 1 round. All N disks can reduce the graph by N degrees in 1 round. The total time is at least $\lfloor \log \frac{N}{\Delta} \rfloor + \frac{2N(\Delta-1)}{N} = \lfloor \log \frac{N}{\Delta} \rfloor + 2(\Delta - 1)$. □

Thus, our solution takes no more than 3 rounds than the optimal.

5 Multi-Source Multicasting

We assume that we have N disks. Disk i, $1 \leq i \leq \Delta \leq N$, has data item i. The goal is to copy item i to a subset D_i of disks that do not have item i. (Hence $i \notin D_i$). In Appendix A we show that finding a schedule with the minimum number of rounds is NP-hard. In this section we present a polynomial time approximation algorithm for this problem. The approximation factor of this algorithm is 4. We also present an improvement that allows the use of bypass nodes.

We define β as $\max_{j=1\ldots N} |\{i | j \in D_i\}|$. In other words, β is an upper bound on the number of different sets D_i to which a disk j may belong. Note that β is a lower bound on the optimal number of rounds, since the disk that attains the maximum, needs at least β rounds to receive all the items i such that $j \in D_i$, since it can receive at most one item in each round.

The algorithm will first create a small number of copies of each data item i (the exact number of copies will be dependent on $|D_i|$). We then assign each newly created copy to a set of disks in D_i, such that it will be responsible for providing item i to those disks. This will be used to construct a transfer graph, where each directed edge labeled i from v to w indicates that disk v must send item i to disk w. We will then use an edge-coloring of this graph to obtain a valid schedule [6]. The main difficulty here is that a disk containing an item is its source, is also the destination for several other data items.

Algorithm Multi-Source Multicast

1. We first compute a disjoint collection of subsets $G_i, i = 1 \ldots \Delta$. Moreover, $G_i \subseteq D_i$ and $|G_i| = \lfloor \frac{|D_i|}{\beta} \rfloor$. (In Lemma 5.1, we will show how such G_i's can be obtained.)
2. Since the G_i's are disjoint, we have the source for item i (namely disk i) send the data to the set G_i using $\lceil \log |D_i| \rceil + 1$ rounds as shown in Lemma 5.2. Note that disk i may itself belong to some set G_j. Let $G'_i = \{i\} \cup G_i$. In other words, G'_i is the set of disks that have item i at the end of this step.
3. We now create a transfer graph as follows. Each disk is a node in the graph. We add directed edges from each disk in G'_i to disks in $D_i \setminus G_i$ such that the out-degree of each node in G'_i is at most $\beta - 1$ and the in-degree of each node in $D_i \setminus G_i$ is 1. (In Lemma 5.3 we show how that this can be done.) This ensures that each disk in D_i receives item i, and that each disk in G'_i does not send out item i to more than $\beta - 1$ disks.

4. We now find an edge coloring of the transfer graph (which is actually a multigraph) and the number of colors used is an upper bound on the number of rounds required to ensure that each disk in D_j gets item j. (In Lemma 5.4 we derive an upper bound on the degree of each vertex in this graph.)

Lemma 5.1. *(Step 1) There is a way to choose disjoint sets G_i for each $i = 1 \dots \Delta$, such that $|G_i| = \lfloor \frac{|D_i|}{\beta} \rfloor$ and $G_i \subseteq D_i$.*

The proof was shown in Lemma 3.2 in [20]. We include it in Appendix B for completeness.

Lemma 5.2. *Step 2 can be done in $\max_i \lceil \log |D_i| \rceil + 1$ rounds.*

Proof. First we assume that $\max_i |D_i| > 2$ and $\beta \geq 2$ since otherwise the problem becomes trivial.

We arbitrarily choose a new source disk s'_i in each G_i and send item i from disk i to s'_i. Because a disk i may send item i to s'_i and receive item j if $i = s'_j$, this initial transfer can take 2 rounds unless the transfer does not make odd cycles (we will consider the case of odd cycles later).

Because sets G_i are disjoint, it then takes $\lceil \log |G_i| \rceil$ rounds to send item i from s'_i to all disks in G_i. The result follows from considering the non-trivial case where $\beta \geq 2$, $\lceil \log |G_i| \rceil \leq \lceil \log \frac{|D_i|}{\beta} \rceil \leq \lceil \log |D_i| - 1 \rceil$.

Now let us consider the case of odd cycles. If any of G_i in the odd cycle is of size at least 2, then we can break the cycle by selecting other disk in G_i as s'_i. Otherwise if the size of all G_i are one, then this step can be done only in 3 rounds (no broadcasting is needed inside G_i) and therefore the lemma is true. □

Lemma 5.3. *We can construct a transfer graph as described in Step 3, such that the in-degree of each node in $D_i \setminus G_i$ from G_i is 1 and the out-degree of each node in G_i is at most $\beta - 1$.*

Proof. We divide each $D_i \setminus G_i$ into disjoint sets $D_{i1}, \dots, D_{im_i}$ where $m_i = \lceil \frac{|D_i|}{\beta} \rceil$ such that $|D_{ij}| = \beta - 1$ for $j = 1, \dots, m_i - 1$ and $|D_{im_i}| = |D_i \setminus G_i| - (\beta - 1)(m_i - 1)$. For each set D_{ij}, we choose a different disk from G'_i and add a directed edge from the disk to all disks in D_{ij}. Because $|D_{ij}| < \beta$ and each disk in $D_i \setminus G_i$ will have an incoming edge from one disk in G'_i, we have a transfer graph as described in Step 3. □

Lemma 5.4. *The in-degree of any disk in the transfer graph is at most β. The out-degree of any disk in the transfer graph is at most $2\beta - 2$. Moreover, the multiplicity of the graph is at most 4.*

Proof. Note that each disk i may belong to at most β sets D_j. Due to its membership in set D_j it may have one incoming edge from some disk in G'_j.

The out-degree of disk i is $\beta - 1$ due to membership in the set G'_i. These are the $\beta - 1$ edges added in Step 3. In addition, i may be in some set G_k (and thus in G'_k); this may cause an extra out-degree of $\beta - 1$. This gives a total out-degree of at most $2\beta - 2$.

Each disk can be a source for two items because it can be the original source of an item i and also belongs to G_k ($k \neq i$). Since the subgraph with edges for only one item is a simple graph, for any pair of disks p, q, there can be two edges from p to q and two more edges in another direction. Therefore, the multiplicity of the transfer graph is at most 4. □

Theorem 5.5. *The total number of rounds required for the multi-source multicast is $\max_i \lceil \log |D_i| \rceil + 3\beta + 3$.*

Proof. Because of Lemma 5.4, we can find an edge coloring of the graph using at most $3\beta + 2$ colors (see Theorem 2.1). Combining with Lemma 5.2, we can finish the multi-source multicast in $\max_i \lceil \log |D_i| \rceil + 3\beta + 3$ rounds. □

Theorem 5.6. *The total number of rounds required for the multi-source multicast problem is at most* $4OPT + 2$.

Proof. Let β_j be $|\{i|j \in D_i\}|$, i.e., the number of different sets D_i, that disk j belongs to. Thus, the in-degree of disk j in any solutions (not using bypass nodes) is β_j. Consider any source disk s_i in the transfer graph as described in Step 3, its total degree is therefore $\beta_{s_i} + (\beta - 1) + (\beta - 1)$. In the optimal solution, the out-degree of any disk s_i must be at least one, since s_i must send its item to some other disk. Thus, $OPT \geq \max_i(\beta_{s_i} + 1)$. The maximum degree of any source disk s_i in the transfer graph is $\max_i \beta_{s_i} + (\beta - 1) + (\beta - 1) \leq OPT + 2\beta - 3$. Consider any disk j which is not the source, its total degree is $\beta_j + (\beta - 1)$. Note that $OPT \geq \max_j \beta_j$ and $\beta \geq 2$, the maximum degree of any non-source disk is $\max_{j \neq s_i} \beta_j + (\beta - 1) = OPT + (\beta - 1) \leq OPT + 2\beta - 3$. Therefore, the maximum degree of the transfer graph is at most $OPT + 2\beta - 3$. We have an algorithm that takes at most $(\max_i \lceil \log |D_i| \rceil + 1) + (OPT + 2\beta - 3) + 4$ rounds. As $\max_i \lceil \log |D_i| \rceil$ and β are also the lower bounds on the optimal number of rounds, the total number of rounds required is at most $4OPT + 2$. □

For the special case in which the source disks are not in any subset D_i, we can develop better bounds.

Corollary 5.7. *When the source disks are not in any subset* D_i*, the total number of rounds required for the multi-source multicast is* $\max_i \lceil \log |D_i| \rceil + 2\beta + 1$.

Proof. Step 2 can be done in $\max_i \lceil \log |D_i| \rceil$ rounds since we can save one round to send item i to s'_i. Also as the original sources do not belong to any G_i, the transfer graph in Step 4 has out-degree at most $\beta - 1$ and multiplicity at most 2. Therefore, the corollary follows. □

Thus we have 3-approximation for this special case.

5.1 $3 + o(1)$-approximation Algorithm

In this section we present a polynomial-time $3+o(1)$-approximation algorithm for the Multi-Source Multicast problem.

In the previous algorithm, each disk only belongs to at most one G_i set. When the size of D_i is small, say $2\beta - 1$, the size of G_i is 1, and the sole disk in G_i is responsible for sending data to $\beta - 1$ disks, while s_i is responsible for sending data to the remaining $\beta - 1$ disks. By allowing a disk to belong to multiple G_i sets, we can decrease the number of disks to which s_i is responsible for sending items. The out-degree of a disk in the transfer graph is reduced, and we can obtain a better bound.

Suppose a disk can now belong to upto p ($\leq \beta$) different G_i sets. In other words, imagine that there are p slots in each disk, and each G_i will occupy exactly $\lfloor p\frac{|D_i|}{\beta} \rfloor$ slots. If G_i occupies a slot in a disk, the disk will be responsible for sending the item to either $\lfloor \frac{\beta}{p} \rfloor - 1$ or $\lceil \frac{\beta}{p} \rceil - 1$ disks in $D_i \setminus G_i$.

Changes to the algorithm

- In Step 1, we create a modified flow network to compute a (not necessarily disjoint) collection of subsets G_i, where $|G_i|$ is $\lfloor p\frac{|D_i|}{\beta} \rfloor$. In addition, each disk belongs to at most p G_i sets. We show in Lemma 5.8 how such G_i's can be obtained.
- In Step 2, although the G_i's are not disjoint, sending items from s_i to G_i is actually another smaller multi-source multicast problem, where β', the upper bound on the number of different destination sets (G_i) to which a disk j in some G_i may belong, is p. Lemma 5.9 describes the details.

– In Step 3, if G_i occupies a slot in a disk j, we would like the disk to satisfy either $\lfloor \frac{\beta}{p} \rfloor - 1$ or $\lceil \frac{\beta}{p} \rceil - 1$ disks in $D_i \setminus G_i$. Moreover, we would like to keep the total out-degree of disk j to be at most $\beta - p$, while disks in G_i together have to satisfy $\lfloor \lfloor p\frac{|D_i|}{\beta} \rfloor (\frac{\beta}{p} - 1) \rfloor$ disks in $D_i \setminus G_i$. We show in Lemma 5.10 how this can be achieved by a network flow computation. We also show the source s_i is responsible for at most $\lceil \frac{\beta}{p} \rceil$ disks.

Lemma 5.8. *In Step 1, there is a way to choose sets G_i for each $i = 1 \ldots \Delta$, such that G_i occupies exactly one slot in each of $\lfloor p\frac{|D_i|}{\beta} \rfloor$ disks, and $G_i \subseteq D_i$. Moreover, each disk has p slots.*

Proof. The basic idea of the proof is similar to that in Lemma 3.2 in Appendix B. First note that we have enough slots for G_i (we have N disks and each disk has p slots).

$$\sum_i \lfloor p\frac{|D_i|}{\beta} \rfloor \leq \frac{p}{\beta} \sum_i |D_i| \leq \frac{p}{\beta}(\beta N) = pN.$$

Now we show how to assign G_i to the slots using network flows. We create a flow network with a source s and a sink t. We also have two sets of vertices U and W. The first set U has Δ nodes, each corresponding to an item. The set W has N nodes, each corresponding to a disk. We add directed edges from s to each node i in U with capacity $\lambda_i = \lfloor p\frac{|D_i|}{\beta} \rfloor$. We add unit capacity edges from node $i \in U$ to $j \in W$ if $j \in D_i$. We also add edges with capacity p from nodes in W to t. We find a max-flow from s to t in this network. We can find a fractional flow of this value as follows: saturate all the outgoing edges from s. From each node i there are $|D_i|$ edges to nodes in W. Send $\lambda_i \frac{1}{|D_i|}$ units of flow along each of the $|D_i|$ outgoing edges from i. Note that since $\lambda_i \frac{1}{|D_i|} \leq \frac{p}{\beta} \leq 1$ this can be done. Observe that the total incoming flow to a vertex in W is at most p since there are at most β incoming edges, each carrying at most $\lambda_i \frac{1}{|D_i|} \leq \frac{p}{\beta}$ units of flow. The min-cut in this network is obtained by simply selecting the outgoing edges from s. An integral max flow in this network will correspond to $|G_i|$ units of flow going from s to i, and from i to a subset of vertices in D_i before reaching t. The vertices to which i has non-zero flow will form the set G_i. The unit capacity edges between U and W ensures that G_i only occupies one slot in each disk, and thus $|G_i|$ is exactly $\lfloor p\frac{|D_i|}{\beta} \rfloor$. □

Lemma 5.9. *Step 2 can be done in $\max_i \log \lfloor p\frac{|D_i|}{\beta} \rfloor + 3p + 4$ steps.*

Proof. Observe that sending items from s_i to G_i is just another smaller multi-source multicast problem. The upper bound on the number of different destination sets (G_i) to which a disk j in some G_i may belong is p. Therefore, using the 4-approximation algorithm described in the previous section, we can send items to all disks in G_i in $(\max_i \log \lfloor p\frac{|D_i|}{\beta} \rfloor + 2) + (p + ((p-1) + (p-1))) + 4 = \max_i \log \lfloor p\frac{|D_i|}{\beta} \rfloor + 3p + 4$ rounds. □

Lemma 5.10. *In Step 3, we can find a transfer graph to satisfy all requests in $D_i \setminus G_i$, where the in-degree is at most β, the out-degree is at most $(\beta - p) + \lceil \frac{\beta}{p} \rceil$, and the multiplicity is at most $2(p + 1)$.*

Proof. To find out how many disks (in $D_i \setminus G_i$) a disk j in G_i should send item i to, while also satisfying the constraints stated in the description of **Changes to the algorithm**, we create a flow network with a source s and a sink t. We also have two sets of vertices U and W. The first set U has Δ nodes, each corresponding to an item. The set W has N nodes, each corresponding to a disk. We add directed edges from s to each node i in U with capacity $\gamma_i = \lfloor \lfloor p\frac{|D_i|}{\beta} \rfloor (\frac{\beta}{p} - 1) \rfloor$. We add edges from node $i \in U$ to $j \in W$ if $j \in G_i$ with capacity $\lceil \frac{\beta}{p} \rceil - 1$. We also add edges with

capacity $\beta - p$ from nodes in W to t. We find a max-flow from s to t in this network. The min-cut in this network is obtained by simply selecting the outgoing edges from s. We can find a fractional flow of this value as follows: saturate all the outgoing edges from s. From each node i there are $|G_i|$ edges to nodes in W. Send $\gamma_i / \lfloor p\frac{|D_i|}{\beta} \rfloor$ units of flow along each of the $|G_i|$ outgoing edges from i. It is easy to see that $\gamma_i / \lfloor p\frac{|D_i|}{\beta} \rfloor \leq \frac{\beta}{p} - 1$, and therefore we do not violate the capacity constraints on edges from U to W. Observe that the total incoming flow to a vertex in W is at most $\beta - p$ since there are at most p incoming edges, each carrying at most $\frac{\beta}{p} - 1$ units of flow. An integral max flow in this network will correspond to γ_i units of flow going from s to i, and from i to all vertices in G_i before reaching t. If f units of flow fare sent from node $i \in U$ to node $j \in W$ means that disk j will send item i to f disks in $D_i \setminus G_i$.

Construct a transfer graph, similar to the method stated in Lemma 5.3, to satisfy all disks in $D_i \setminus G_i$. As in Lemma 5.4, the in-degree of this transfer graph is at most β. For each disk which belongs to some G_i, its out-degree is at most $\beta - p$. Among all disks in D_i, $\lfloor p\frac{|D_i|}{\beta} \rfloor$ disks are satisfied in Step 2 since they belong to G_i, and G_i can satisfy $\lfloor \lfloor p\frac{|D_i|}{\beta} \rfloor (\frac{\beta}{p} - 1) \rfloor$ disks in Step 3. The number of disks that still need item i are:

$$|D_i| - \lfloor p\frac{|D_i|}{\beta} \rfloor - \lfloor \lfloor p\frac{|D_i|}{\beta} \rfloor (\frac{\beta}{p} - 1) \rfloor = |D_i| - \lfloor \lfloor p\frac{|D_i|}{\beta} \rfloor \frac{\beta}{p} \rfloor \leq |D_i| - \lfloor |D_i| - \lceil \frac{\beta}{p} \rceil \rfloor = \lceil \frac{\beta}{p} \rceil.$$

Source s_i is responsible for all these disks. Therefore the out-degree of s_i is at most $\lceil \frac{\beta}{p} \rceil$, and the total out-degree of a node is at most $(\beta - p) + \lceil \frac{\beta}{p} \rceil$.

Similar to Lemma 5.4, each disk can be a source for up to $p + 1$ items, because it can be the original source of item i, and it also belongs to p different G_k $(k \neq i)$ sets. Thus there are upto $p + 1$ directed edges in each direction. □

Theorem 5.11. *The total number of rounds is* $\max_i \log \lfloor p\frac{|D_i|}{\beta} \rfloor + 2\beta + \lceil \frac{\beta}{p} \rceil + 4p + 6$. *When* p *is* $\Theta(\sqrt{\beta})$, *the total number of rounds is minimized, and is equal to* $\max_i \log |D_i| + 2\beta + O(\sqrt{\beta})$.

Proof. The number of rounds taken in Step 3 is $2\beta + \lceil \frac{\beta}{p} \rceil + p + 2$ from Lemma 5.10 and Theorem 2.1. Combined with Lemma 5.9, the first result can be easily obtained. The second result is obtained by substituting p with $\Theta(\sqrt{\beta})$. □

As $\max_i \log |D_i|$ and β are lower bounds of the problem, from Theorem 5.11, we have a polynomial-time $3 + o(1)$-approximation algorithm.

5.2 Allowing Bypass Nodes

The main idea is that without bypass nodes, only a small fraction of N disks is included in G_i for some i, if one disk requests many items while, on average, each disk requests few items. If we allow bypass nodes and hence G_i is not necessarily a subset of D_i, we can make G_i very big so that each of almost all N disks belongs to some G_i. Bigger G_i reduces the out-degree of the transfer graphs and thus reduces the total number of rounds.

Algorithm Multi-Source Multicast Allowing Bypass Nodes

1. We define $\overline{\beta}$ as $\frac{1}{N} \sum_{i=1 \ldots N} |\{j | i \in D_j\}|$. In other words, $\overline{\beta}$ is the number of items a disk could receive, averaging over all disks. We arbitrarily choose a disjoint collection of subsets G_i, $i = 1 \ldots \Delta$ with a constraint that $|G_i| = \lfloor \frac{|D_i|}{\lceil \overline{\beta} \rceil} \rfloor$. By allowing bypass nodes, G_i is not necessarily a subset of D_i.
2. This is the same as Step 2 in the Multi-Source Multicast Algorithm, except that the source for item i (namely disk i) may belong to G_j for some j.

3. This step is similar to Step 3 in the Multi-Source Multicast Algorithm. We add $\lceil\overline{\beta}\rceil$ edges from each disk in G_i to satisfy $\lceil\overline{\beta}\rceil \cdot \lfloor\frac{|D_i|}{\lceil\overline{\beta}\rceil}\rfloor$ disks in D_i, and add at most another $\lceil\overline{\beta}\rceil - 1$ edges from disk i to satisfy the remaining disks in D_i.
4. This is the same as Step 4 in the Multi-Source Multicast Algorithm.

Theorem 5.12. *The total number of rounds required for the multi-source multicast algorithm, by allowing bypass nodes, is* $\max_i\lceil\log|D_i|\rceil + \beta + \lceil 2\overline{\beta}\rceil + 6$.

Proof. The analysis is very similar to the case without bypass nodes and here we only highlight the differences. Note that the total size of the sets G_i is at most N.

$$\sum_i |G_i| \leq \sum_i \frac{|D_i|}{\lceil\overline{\beta}\rceil} \leq \frac{1}{\overline{\beta}} \sum_i |D_i|.$$

Note that $\sum_i |D_i|$ is $\overline{\beta}N$ by the definition of $\overline{\beta}$. This proves the upper bound of N on the total size of all the sets G_i. Step 2 takes $\max_i\lceil\log|D_i|\rceil + 2$ rounds. Note that this is 1 round larger than the bound in Lemma 5.2 as $\lceil\overline{\beta}\rceil$ can be 1. The in-degree of any disk in the transfer graph is still at most β, while the out-degree of any disk in the transfer graph is at most $\lceil\overline{\beta}\rceil + (\lceil\overline{\beta}\rceil - 1)$. The multiplicity of the graph is still at most 4. Thus, the total number of rounds is $(\max_i\lceil\log|D_i|\rceil + 2) + \beta + \lceil\overline{\beta}\rceil + (\lceil\overline{\beta}\rceil - 1) + 4 \leq \max_i\lceil\log|D_i|\rceil + \beta + \lceil 2\overline{\beta}\rceil + 6$. □

We now argue that $\lceil 2\overline{\beta}\rceil$ is a lower bound on the optimal number of rounds. Intuitively, on average, every disk has to spend $\overline{\beta}$ rounds to send data, and another $\overline{\beta}$ rounds to receive data. As a result, the total number of rounds cannot be smaller than $\lceil 2\overline{\beta}\rceil$. This can be seen by simply computing the total number of required transfers, and dividing by the number of transfers that can take place in each round. Allowing bypass node does not change the fact that $\max(\max_i\lceil\log|D_i|\rceil, \beta)$ is the other lower bound. Therefore, we have a 3-approximation algorithm.

References

1. E. Anderson, J. Hall, J. Hartline, M. Hobbes, A. Karlin, J. Saia, R. Swaminathan and J. Wilkes. An Experimental Study of Data Migration Algorithms. *Workshop on Algorithm Engineering*, 2001
2. B. Baker and R. Shostak. Gossips and Telephones. *Discrete Mathematics*, 2:191–193, 1972.
3. J. Bermond, L. Gargano and S. Perennes. Optimal Sequential Gossiping by Short Messages. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, Vol 86, 1998.
4. J. Bermond, L. Gargano, A. A. Rescigno and U. Vaccaro. Fast gossiping by short messages. *International Colloquium on Automata, Languages and Programming*, 1995.
5. S. Berson, S. Ghandeharizadeh, R. R. Muntz, and X. Ju. Staggered Striping in Multimedia Information Systems. *SIGMOD*, 1994.
6. J. A. Bondy and U. S. R. Murty. Graph Theory with applications. *American Elsevier*, New York, 1977.
7. R. T. Bumby. A Problem with Telephones. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):13–18, March 1981.
8. E. J. Cockayne, A. G. Thomason. Optimal Multi-message Broadcasting in Complete Graphs. *Utilitas Mathematica*, 18:181–199, 1980.
9. G. De Marco, L. Gargano and U. Vaccaro. Concurrent Multicast in Weighted Networks. *SWAT*, 193–204, 1998.
10. A. M. Farley. Broadcast Time in Communication Networks. *SIAM Journal on Applied Mathematics*, 39(2):385–390, 1980.
11. P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, 53:79–133, 1994.

12. L. Golubchik, S. Khanna, S. Khuller, R. Thurimella and A. Zhu. Approximation Algorithms for Data Placement on Parallel Disks. *Proc. of ACM-SIAM SODA*, 2000.
13. J. Hall, J. Hartline, A. Karlin, J. Saia and J. Wilkes. On Algorithms for Efficient Data Migration. *Proc. of ACM-SIAM SODA*, 620–629, 2001.
14. A. Hajnal, E. C. Milner and E. Szemeredi. A Cure for the Telephone Disease. *Canadian Mathematical Bulletin*, 15(3):447–450, 1972.
15. S. M. Hedetniemi, S. T. Hedetniemi and A. Liestman. A Survey of Gossiping and Broadcasting in Communication Networks. *Networks*, 18:129–134, 1988.
16. I. Holyer. The NP-Completeness of Edge-Coloring. *SIAM J. on Computing*, 10(4):718–720, 1981.
17. J. Hromkovic and R. Klasing and B. Monien and R. Peine. Dissemination of Information in Interconnection Networks (Broadcasting and Gossiping). *Combinatorial Network Theory*, pp. 125–212, D.-Z. Du and D.F. Hsu (Eds.), Kluwer Academic Publishers, Netherlands, 1996.
18. C. A. J. Hurkens. Spreading Gossip Efficiently. *Nieuw Archief voor Wiskunde*, 5(1):208–210, 2000.
19. S. Kashyap and S. Khuller. Algorithms for Non-Uniform Size Data Placement on Parallel Disks. *Manuscript*, 2003.
20. S. Khuller, Y. A. Kim and Y. C. Wan. Algorithms for Data Migration with Cloning. To appear, ACM Symp. on Principles of Database Systems (2003).
21. W. Knodel. New gossips and telephones. *Discrete Mathematics*, 13:95, 1975.
22. H. M. Lee and G. J. Chang. Set to Set Broadcasting in Communication Networks. *Discrete Applied Mathematics*, 40:411–421, 1992.
23. D. Liben-Nowell. Gossip is Synteny: Incomplete Gossip and an Exact Algorithm for Syntenic Distance. *Proc. of ACM-SIAM SODA*, 177–185, 2001.
24. C. H. Papadimitriou. Computational complexity. *Addison-Wesley*, 1994.
25. D. Richards and A. L. Liestman. Generalizations of Broadcasting and Gossiping. *Networks*, 18:125–138, 1988.
26. H. Shachnai and T. Tamir. On two class-constrained versions of the multiple knapsack problem. *Algorithmica*, 29:442–467, 2001.
27. H. Shachnai and T. Tamir. Polynomial time approximation schemes for class-constrained packing problems. *Proc. of Workshop on Approximation Algorithms*, 2000.
28. C.E. Shannon. A theorem on colouring lines of a network. *J. Math. Phys.*, 28:148–151, 1949.
29. M. Stonebraker. A Case for Shared Nothing. *Database Engineering*, 9(1), 1986.
30. R. Tijdeman. On a Telephone Problem. *Nieuw Archief voor Wiskunde*, 19(3):188–192, 1971.
31. V. G. Vizing. On an estimate of the chromatic class of a p-graph (Russian). *Diskret. Analiz.* 3:25–30, 1964.

Appendices

A NP-hardness

We will prove the multi-source multicasting problem is NP-hard by showing a reduction from a restricted version of 3SAT. Papadimitriou [24] showed that 3SAT remains NP-complete even for expressions in which each variable is restricted to appear at most three times, and each literal at most twice. We denote this problem as 3SAT(3).

We assume that each literal appear at least once in the given instance. If not, we can always simplify it such that all literal appear at least once (or the instance is always true).

Given a 3SAT(3) instance, we create a multi-source multicast instance such that the 3SAT(3) instance is satisfied if and only if the corresponding multi-source multicast instance can transfer all items in 3 rounds.

Part I. For each variable x_i, we create (i) a source disk having item x_i, (ii) a set of destination disks X_i of size 3 which need item x_i, (iii) a source disk having item $\overline{x}_i$, (iv) a set of destination disks $\overline{X}_i$ of size 3 which need item $\overline{x}_i$, (v) a source disk having item s_i, (vi) a disk w_i (we call it a switch disk) which wants to receive items x_i, $\overline{x}_i$ and s_i, and (vii) 6 disks which need item s_i.

Part II. For each clause j, we create (i) a source disk having item c_j, and (ii) a set of destination disks C_j of size 2 which need item c_j. Moreover, for each literal in the clause j, arbitrarily pick one disk in the set of destination disks corresponding to the literal, and that disk, which originally only need the item corresponding to the literal, will also need item c_j. For example, if clause j is $x_p \vee \overline{x}_q \vee x_r$, then one disk d in X_p, one disk in $\overline{X}_q$ and one disk in X_r, need item c_j. If there is another clause j' contains literal x_p, we pick one disk in $X_p \setminus \{d\}$ and that disk now needs item j'.

Lemma A.1. *If the 3SAT(3) instance is satisfiable, there exists a valid schedule to finish all data transfers in* 3 *rounds.*

Proof. It is easy to see that all seven disks demanding item s_i can be scheduled in three rounds. In particular, we schedule switch disk w_i to receive s_i in round 3 for all i. If variable x_i is **true**, we schedule switch disk w_i to receive $\overline{x}_i$ and x_i in round 1 and 2 respectively. x_i can be sent to a disk in X_i in round 1, making X_i receive items faster than $\overline{X}_i$. After round 2, 2 disks in X_i received item x_i, while only 1 disk in $\overline{X}_i$ received item $\overline{x}_i$. In round 3, the source disk of x_i can satisfy the last disk in X_i which has not received x_i. Note that the remaining 2 disks in X_i are idle and they can receive item c_j from other disks. On the other hand, the remaining 2 disks in $\overline{X}_i$ can be satisfied in round 3 by the source and one disk in $\overline{X}_i$. Note that all disks in $\overline{X}_i$ and the source of $\overline{x}_i$ are busy in this round. Thus, all requested items appeared in **Part I** are satisfied. If the variable is **false**, we schedule the switch disk to receive x_i in round 1, then $\overline{x}_i$ in round 2. As a result, 2 disks in $\overline{X}_i$ are idle in round 3, while all disks in X_i are busy in round 3.

We claim that all 2 disks in C_j, for all j, can be satisfied too. For example, if clause j is $x_p \vee \overline{x}_q \vee x_r$, and suppose x_p is **true** while $\overline{x}_q$ and x_r are **false** in a satisfactory assignment. From the argument above, there exists a schedule such that the disk in X_p, which needs x_p and c_j in X_p, is idle in round 3. However, the disk in $\overline{X}_q$, which needs $\overline{x}_q$ and c_j, and the disk in X_r, which needs x_r and c_j, are busy in round 3. A valid schedule can send item c_j from the source to one disk in C_j in round 1. In round 2, we now have 2 copies of c_j to satisfy disks in $\overline{X}_q$ and X_r. In round 3, without the help of disks in $\overline{X}_q$ and X_r, we can satisfy 2 more disks, which are the second disk in C_j and the disk in X_p. Thus, all requested items appeared in **Part II** are satisfied too. □

Lemma A.2. *If there is a valid schedule to finish all data transfers in 3 rounds, then the 3SAT(3) instance is satisfiable.*

Proof. Since there are 7 disks need item s_i, if we have to finish all transfers in 3 rounds, once a disk receives s_i, it will become busy until round 3. Note that all switch disks have to receive s_i, x_i and $\overline{x}_i$. All switch disk have to receive item x_i and $\overline{x}_i$ in the first two rounds, and s_i in the round 3. If switch disk i receives item x_i in round 1, we set literal $\overline{x}_i$ to be **true**. Otherwise, we set literal x_i to be **true**. Consider the former case, disks in X_i receive item x_i starting at round 2, meaning that all disks in X_i should be busy in round 3 to send or receive x_i. Suppose literal x_i appears in clause j and k. 2 disks in X_i have to receive item c_j and c_k in the first 2 rounds. Thus, our construction restricts that if a literal x_i is set to **false**, disks in X_i cannot receive item c_j in round 3.

Consider a clause j, for instance, $x_p \vee \overline{x}_q \vee x_r$, a disk in X_p, a disk in $\overline{X}_q$, a disk in X_r, and all 2 disks in C_j need item c_j. If all three literals are **false**, it is possible to satisfy the first three disks in the first 2 rounds. However, since all these three disks are busy in round 3, the source of c_j cannot satisfy both disks in C_j, which is a contradiction. Therefore, the clause j should be satisfied. □

Theorem A.3. *The multi-source multicasting problem is NP-hard*

Proof. It is easy to see that the reduction is polynomial, together with Lemma A.1 and Lemma A.2, the problem is NP-hard. □

B Proof of Lemma 3.2 in [20]

We include the proof here for completeness purposes.

Lemma 3.2. (Step 1) There is a way to choose disjoint sets G_i for each $i = 1 \dots \Delta$, such that $|G_i| = \lfloor \frac{|D_i|}{\beta} \rfloor$ and $G_i \subseteq D_i$.

Proof. First note that the total size of the sets G_i is at most N.

$$\sum_i |G_i| \leq \sum_i \frac{|D_i|}{\beta} = \frac{1}{\beta} \sum_i |D_i|.$$

Note that $\sum_i |D_i|$ is at most βN by definition of β. This proves the upper bound of N on the total size of all the sets G_i.

We now show how to find the sets G_i. We create a flow network with a source s and a sink t. In addition we have two sets of vertices U and W. The first set U has Δ nodes, each corresponding to a disk that is the source of an item. The set W has N nodes, each corresponding to a disk in the system. We add directed edges from s to each node in U, such that the edge (s, i) has capacity $\lfloor \frac{|D_i|}{\beta} \rfloor$. We also add directed edges with infinite capacity from node $i \in U$ to $j \in W$ if $j \in D_i$. We add unit capacity edges from nodes in W to t. We find a max-flow from s to t in this network. The min-cut in this network is obtained by simply selecting the outgoing edges from s. We can find a fractional flow of this value as follows: saturate all the outgoing edges from s. From each node i there are $|D_i|$ edges to nodes in W. Suppose $\lambda_i = \lfloor \frac{|D_i|}{\beta} \rfloor$. Send $\frac{1}{\beta}$ units of flow along $\lambda_i \beta$ outgoing edges from i. Note that since $\lambda_i \beta \leq |D_i|$ this can be done. Observe that the total incoming flow to a vertex in W is at most 1 since there are at most β incoming edges, each carrying at most $\frac{1}{\beta}$ units of flow. An integral max flow in this network will correspond to $|G_i|$ units of flow going from s to i, and from i to a subset of vertices in D_i before reaching t. The vertices to which i has non-zero flow will form the set G_i. □

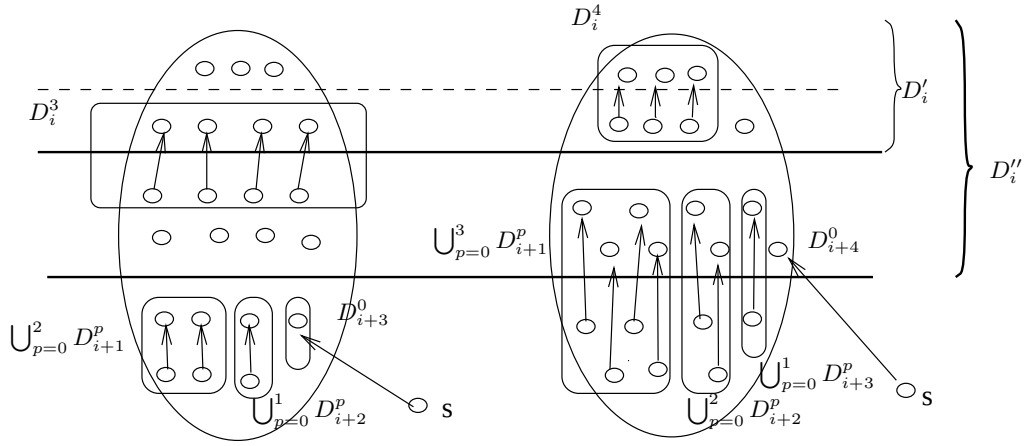

Fig. 4. The figure shows how disks in D_i behave in Phase I where $|D_i| = 2^4 + 2^2 + 2^1$

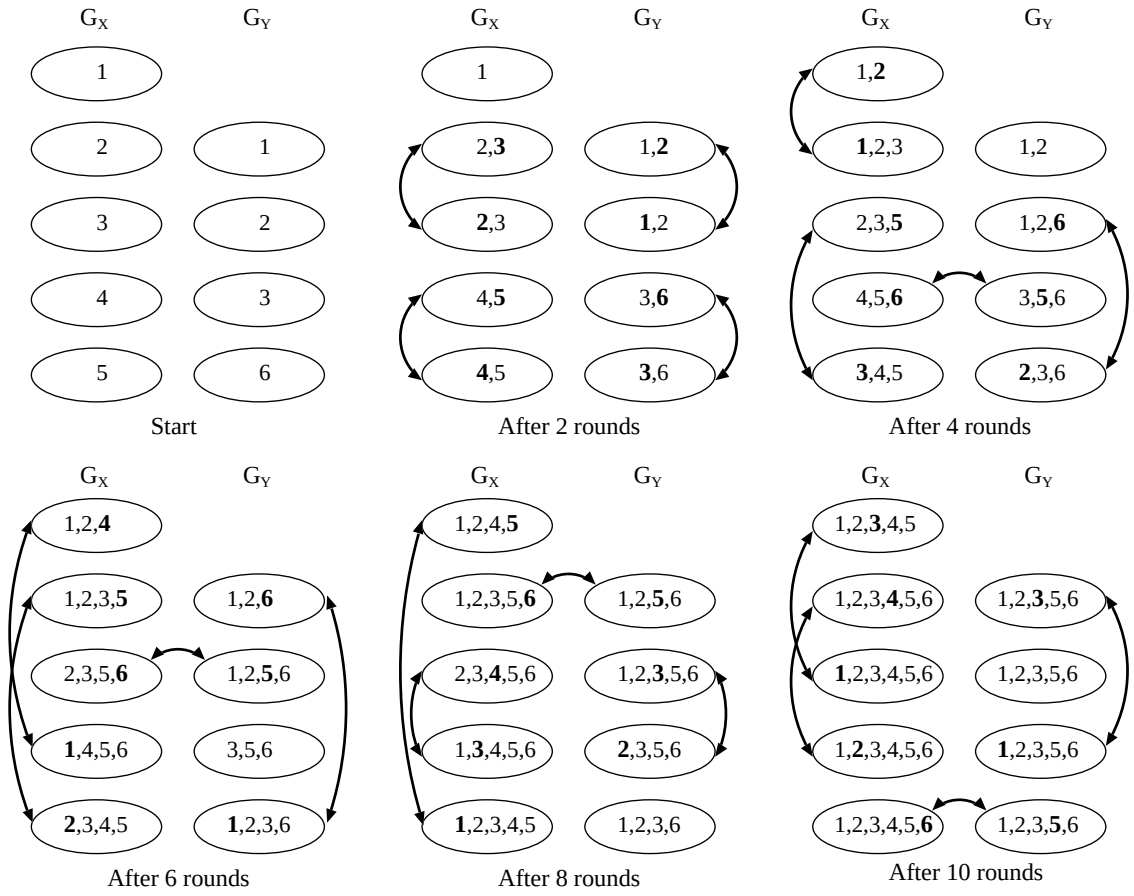

Fig. 5. An example of Case III in Multi-Source Broadcasting section with $\Delta = 6$ and $r = 3$. Recently received items are in bold