Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Algorithms 59 (2006) 81–106

Journal of Algorithms

www.elsevier.com/locate/jalgor

On generalized gossiping and broadcasting ☆

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

Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA

Received 3 June 2003

Available online 30 March 2005

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.

☆ This research was supported by NSF Awards CCR-9820965 and CCR-0113192.

* Corresponding author.

E-mail addresses: samir@cs.umd.edu (S. Khuller), ykim@cs.umd.edu (Y.-A. Kim), ycwan@cs.umd.edu (Y.-C. Wan).

0196-6774/$ – see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jalgor.2005.01.002

1. Introduction

The problems of gossiping and broadcasting have been the subject of extensive study [3,4,15–17,20]. 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 [22,24]. (In Section 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 [5,27]. 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 [12,18,25,26] 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

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

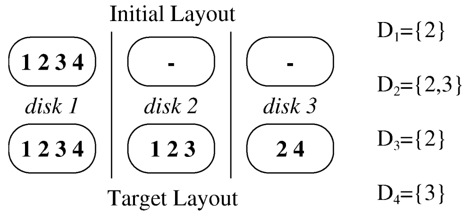


Fig. 1. Initial and target layouts, and their corresponding D_i's for a single-source multicast instance.

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 [1,13] 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 previously by Cockayne, Thomason and Farley [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.

1.1. 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* $OPT + 3$. *In particular the number of rounds needed is* $\lceil \log(N/\Delta) \rceil + 2\Delta$.

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. Bypass disks can be used to hold data temporarily. For example, a source for item i may send i to a bypass disk, which later forwards the item to a disk in D_i even though the bypass disk itself is not in D_i. Thus we use no bypass (intermediate) disks as holding points for the data. If bypass disks are allowed, we have the following result:

Theorem 1.5. *For the multi-source multicast problem allowing bypass disks 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 [19]. 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 developed in [19] are not as good as the bounds we can get for the specific problems. The methods used for single-source multicast and multi-source broadcast are completely different from the algorithm in [19]. Using the methods in [19] one cannot obtain additive bounds from the optimal solution. The algorithm for multi-source multicast presented here is a simplification of the general algorithm developed [19], but we also obtain a much better approximation factor of 4. In addition, by allowing bypass disks we can improve the bounds further. In addition, by using new ideas we can improve it to $3 + o(1)$ without using bypass disks.

Many generalizations of gossiping and broadcasting have been studied before. For example, the paper by Liben-Nowell [22] 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 [2,7,14,17,28]. Liben-Nowell [22] gives an exponential time exact algorithm for the problem.

Other related problems that have been studied are the set-to-set gossiping problem [21,24] 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 Lee and Chang [21] 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 [22] 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 previously [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 disks, 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 [29])). If G has no self-loops then $\chi' \leqslant \Delta_G + \mu$.*

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 $\lceil (\Delta(N-1) - 2^{\lfloor \log_2 N \rfloor} + 1)/\lfloor N/2 \rfloor \rceil + \lfloor \log N \rfloor$ 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

Our algorithm consists of two phases. In the first phase, we make exactly $\lfloor |D_i|/2 \rfloor$ copies for all items i. Once each item i has $\lfloor |D_i|/2 \rfloor$ copies, in second phase we can finish migrating one item at each round by copying from the current copies to the remaining $\lfloor |D_i|/2 \rfloor$ disks in D_i which have not received item i as yet and using the source disk to make another copy if $|D_i|$ is odd.

It is easy to see that the second phase can be scheduled without conflicts as we deal with only one item in each round. For the first phase, let us consider the simple example shown in Fig. 2. In this example, all D_i are identical and include all disks (thus the problem is the same as broadcasting [8,10]) and $\Delta = 4$, $|D_i| = 12$ for each item i. At each round, the source disk makes a new copy. For other items, the numbers of copies are doubled if

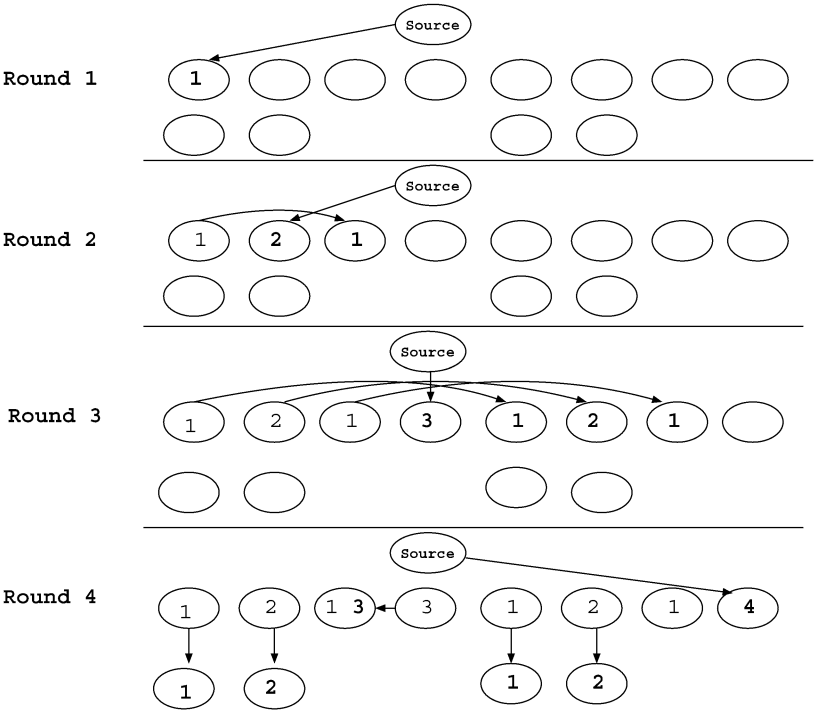


Fig. 2. An example when all D_i are the same and $\Delta = 4$, $|D_i| = 12$ for all item i.

possible. Consider Round 4. Since there are four copies of item 1, only two copies need to be created to make $|D_i|/2 = 6$ copies. For items 2 and 3, we can double the number of copies, and a new copy for item 4 is created by the source disk.

Without loss of generality, we assume that $|D_1| \geqslant |D_2| \geqslant \cdots \geqslant |D_\Delta|$ (otherwise renumber the items). Let d_i be the largest index such that $2^{d_i} \leqslant |D_i|$. For example, if $|D_i| = 12$, then $d_i = 3$.

Phase I. At the tth round, we do the following.

1. The source disk s creates a new copy for item t if $t \leqslant \Delta$.
2. For items j $(j < t)$, double the number of copies until the number of copies becomes $\lfloor |D_j|/2 \rfloor$. In other words, if the current number of copies of item j is less than or

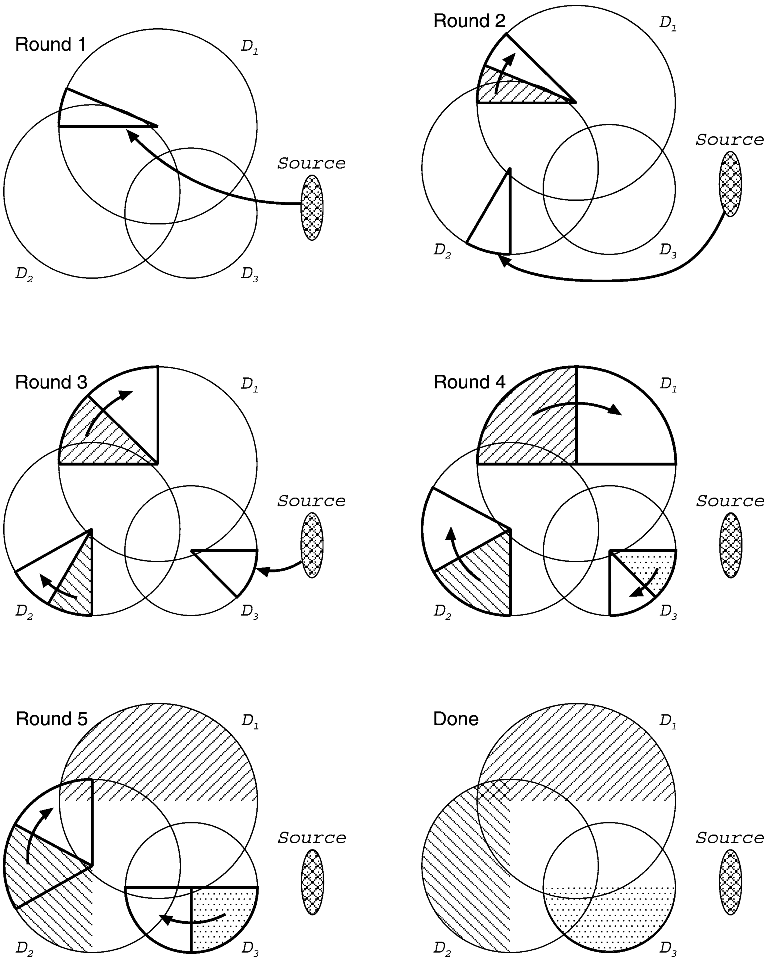


Fig. 3. An example of Phase I when all $|D_i|$ are even.

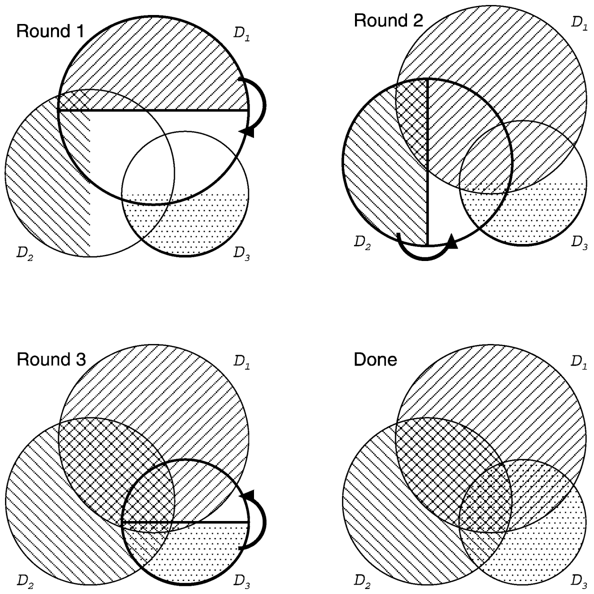


Fig. 4. An example of Phase II when all $|D_i|$ are even.

equal to 2^{d_j-2}, every disk having item j makes another copy of it so that the number of copies is doubled. Otherwise if the current number of copies of item j is 2^{d_j-1}, then only $\lfloor |D_j|/2\rfloor - 2^{d_j-1}$ disks need to make copies (thus the number of copies of item j becomes exactly $\lfloor |D_j|/2\rfloor$).

Phase II. After Phase I, each item j has exactly $\lfloor |D_j|/2\rfloor$ copies. Therefore, at each round, we finish the migration of one item. At the tth round, we copy item t 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 to make one more copy if $|D_t|$ is odd.

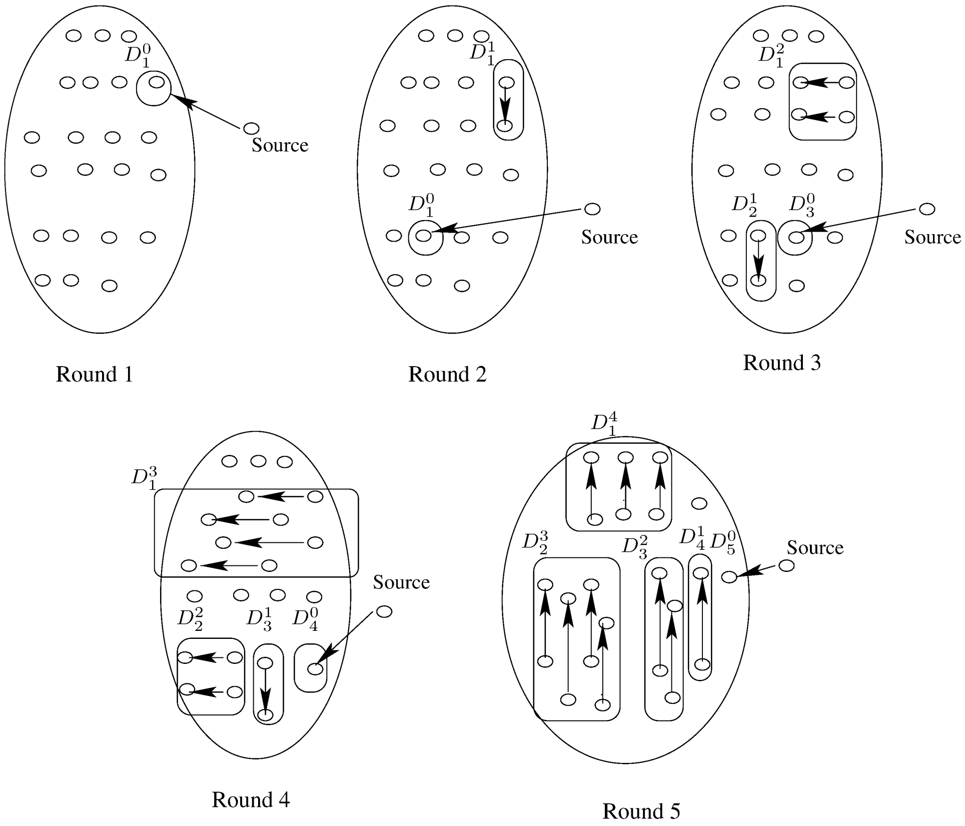
Figures 3 and 4 show an example of data transfers in Phases I and II, where $|D_1|$, $|D_2|$ and $|D_3|$ are 16, 12 and 8, respectively.

Since migrations of several items happen at the same time in Phase I, and D_i's are arbitrary, we need to carefully choose which subset of disks will participate in the migration of each item. We explain the details of how we can perform Phase I without conflicts in the next section.

3.2. *Details of Phase I*

Recall that we assume that $|D_1| \geqslant |D_2| \geqslant \cdots \geqslant |D_\Delta|$ and make copies starting from $D_1, D_2, \ldots$. 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. Then the size of D_i^p should be

$$|D_i^p| = \begin{cases} 2^p & \text{if } p \leqslant d_i - 1, \\ 2(\lfloor |D_i|/2\rfloor - 2^{d_i-1}) & \text{if } p = d_i. \end{cases}$$



Round 1 Round 2 Round 3

Round 4 Round 5

Fig. 5. The figure shows how disks in D_i behave in Phase I where $|D_i| = 2^4 + 2^2 + 2^1$.

D_i^0 is the first disk receiving i from the source s, and the size is doubled at each round until the number of copies becomes $\lfloor |D_i|/2 \rfloor$. Figure 5 shows the migration in Phase I.

We build D_Δ^p first as follows. For $D_\Delta^{d_\Delta}$, choose $2(\lfloor |D_\Delta|/2 \rfloor - 2^{d_\Delta - 1})$ disks arbitrarily from D_Δ. When we choose $D_\Delta^{d_\Delta - 1}$, choose $\lfloor |D_\Delta|/2 \rfloor - 2^{d_\Delta - 1}$ disks from $D_\Delta^{d_\Delta}$ and choose $2^{d_\Delta} - \lfloor |D_\Delta|/2 \rfloor$ disks from $D_\Delta \setminus D_\Delta^{d_\Delta}$ so that the size of $D_\Delta^{d_\Delta - 1}$ is $2^{d_\Delta - 1}$. For each D_Δ^p ($p < d_\Delta - 1$), choose 2^p disks from D_Δ^{p+1}. Note that for all p, exactly half of disks in D_Δ^p are included in D_Δ^{p-1} (which will be used as senders) and the remaining half is not included (which will be used as receivers). For example, if the size of D_Δ is 12, then we choose 1, 2, 4 disks for D_Δ^p ($p = 0, 1, 2$) respectively and for D_Δ^3, include 2 disks from D_Δ^2 and 2 disks from $D_\Delta \setminus D_\Delta^2$.

We now decide D_i^p, given all $D_j^{p'}$ ($j > i$). At ($i + p$)th round (when disks in D_i^p participate in migration for item i), disks in D_j^{i+p-j} for all j ($i < j \leqslant \min(i + p, \Delta)$) also participate in either sending or receiving item j at the same time. We have to decide which disks belong to D_i^p to avoid conflicts with D_j^{i+p-j}'s ($j > i$).

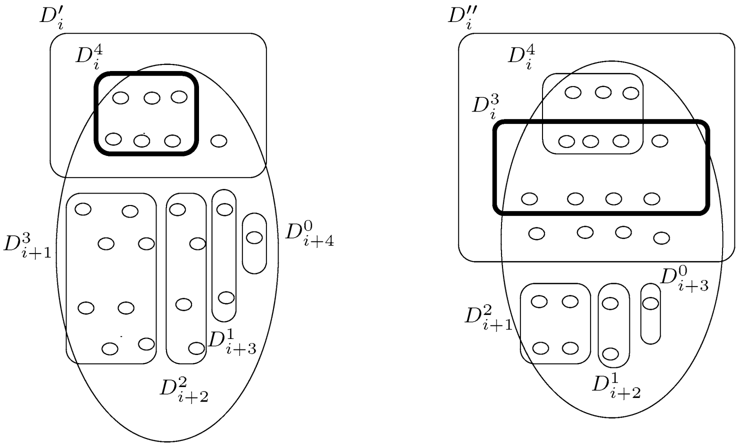


Fig. 6. The figure shows how to choose disks in D_i^p for Phase I where $|D_i| = 2^4 + 2^2 + 2^1$.

Consider $D_i^{d_i}$. The set should not be overlapped with any set $D_j^{i+d_i-j}$ $(j > i)$ since they also participate in migration at $(i + d_i)$th round. Therefore we define

$$D_i' = D_i - \bigcup_{j=i+1}^{\Delta} D_j^{i+d_i-j}$$

and choose $D_i^{d_i}$ from D_i'. Similarly, set $D_i^{d_i-1}$ should not be overlapped with any set $D_j^{i+d_i-1-j}$, $j > i$. Therefore, we define

$$D_i'' = D_i - \bigcup_{j=i+1}^{\Delta} D_j^{i+d_i-1-j}$$

and choose set $D_i^{d_i-1}$ from D_i''. Figure 6 shows how to choose D_i^p. Note that half of $D_i^{d_i}$ should be included in $D_i^{d_i-1}$ (to be senders) and the remaining half should be excluded from $D_i^{d_i-1}$ (to be receivers). For D_i^p $(p < d_i - 1)$, we can choose half the disks from D_i^{p+1}.

Lemma 3.1. *We can find a migration schedule in 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 $|D_j^p| \leqslant 2^p$,

$$|D_i''| = \left| D_i - \bigcup_{j=i+1}^{\Delta} D_j^{i+d_i-1-j} \right|$$

$$\geqslant |D_i| - \sum_{j=i+1}^{\Delta} 2^{i+d_i-1-j}$$
$$\geqslant |D_i| - \sum_{m=0}^{d_i-2} 2^m > |D_i| - 2^{d_i-1}.$$

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

$$\left|D'_i\right| = |D_i - \bigcup_{j=i+1}^{\Delta} D_j^{i+d_i-j}| > |D_i| - 2^{d_i}.$$

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

Now we argue that there is no conflict in performing migration if we do migration according to D_i^p. Since $D_i^{d_i} \subset D'_i$ and $D'_i \cap D_j^{i+d_i-j} = \emptyset$ $(j > i)$, there is no conflict between i and j at $(i + d_i)$th round. For $p \leqslant d_i - 1$, since $D_i^p \subset D''_i$ and $D''_i \cap D_j^{i+p-j} = \emptyset$ $(j > i)$, 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, $C(I) \geqslant \max_{1 \leqslant i \leqslant \Delta}(i + \lfloor \log |D_i| \rfloor)$.*

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, $C(I) \geqslant \max_{1 \leqslant i \leqslant \Delta}(f(i) + \lfloor \log |D_i| \rfloor)$ 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 $\max_{1 \leqslant i \leqslant \Delta}(f(i) + \lfloor \log |D_i| \rfloor) \geqslant \max_{1 \leqslant i \leqslant \Delta}(i + \lfloor \log |D_i| \rfloor)$ for any permutation $f(i)$. Suppose there is a set D_i that $f(i) \neq i$ when $\max_{1 \leqslant i \leqslant \Delta}(f(i) + \lfloor \log |D_i| \rfloor)$ 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

$$\max\bigl(f(i) + \lfloor \log |D_i| \rfloor, f(j) + \lfloor \log |D_j| \rfloor\bigr) = f(j) + \lfloor \log |D_j| \rfloor$$
$$\geqslant \max\bigl(j + \lfloor \log |D_j| \rfloor, f(j) + \lfloor \log |D_i| \rfloor\bigr).$$

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

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

Proof. In Phase I, D_i receives i from s at ith round for the first time. Because the number of copies doubles 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 Lemmas 3.2 and 3.3. □

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

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

4. Multi-source broadcasting

We assume that we have N disks. Disk i, $1 \leqslant i \leqslant \Delta$, has item numbered i. The goal is to send each item i to all N disks, for all i. We present an algorithm that takes at most 3 more rounds than the optimal solution.

4.1. Algorithm Multi-Source Broadcast

For the high-level description we assume for simplicity that N is a multiple of Δ. The main idea behind the algorithm is the following (as shown in Fig. 7). We first partition the

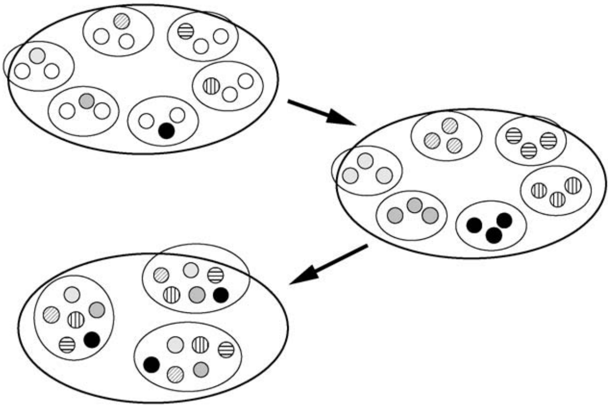


Fig. 7. An example to illustrate the main idea behind algorithm Multi-Source Broadcast ($N = 18$ and $\Delta = 6$).

disks into Δ equal sized groups $G_1, \ldots, G_\Delta$. Group G_i contains the source disk of item i. We now perform a broadcast within each group so that each disk in group G_i contains item i. We now make N/Δ new groups of size Δ by picking one disk from each group. Since each disk contains a different item, this is exactly the gossip problem for a set of Δ disks. We solve all these gossip problems in parallel. Now all disks contain all the items. The actual algorithm is a little more complicated since it works for arbitrary values of N.

1. We partition the N disks into Δ sets G_i such that disk $i \in G_i$, for all $i = 1, \ldots, \Delta$. Let q be $\lfloor N/\Delta \rfloor$ and r be $N - q\Delta$. $|G_i| = q + 1$ for $i = 1, \ldots, r$, and $|G_i| = q$ for $i = r + 1, \ldots, \Delta$. We do a broadcast in group G_i of item i. This takes $\lceil \log |G_i| \rceil$ rounds by doubling the number of items in each round. (Each disk that receives an item, sends it out in each subsequent round until all the disks in the group have the item.)
2. We now partition the N disks into $q - 1$ groups of size Δ each, by picking one disk from each G_i, and one group of size $\Delta + r$ that contains all the remaining disks.
3. Consider the first $q - 1$ 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 receives all Δ items in 2Δ rounds.[2]
4. In the last gossiping group, for each of the items numbered $1, \ldots, r$, there are exactly two disks having this item. For each of the items numbered $r + 1, \ldots, \Delta$, there is exactly one disk having this item. Note that each disk has exactly one item. If r is zero, we can finish all transfers in 2Δ rounds using the algorithm in [4]. For non-zero r, we claim that all disks in this gossiping group still receive all items in 2Δ rounds.
   We divide the disks in the last gossiping group into two groups, G_X and G_Y of size $\Delta - \lfloor (\Delta - r)/2 \rfloor$ and $r + \lfloor (\Delta - r)/2 \rfloor$ respectively, as shown in Fig. 8. Note that $|G_Y| + 1 \geqslant |G_X| \geqslant |G_Y|$. Each of the items numbered $1, \ldots, r$ appear in both G_X and G_Y; disks having items $r + 1, \ldots, \Delta - \lfloor (\Delta - r)/2 \rfloor$ are in G_X, and the remaining disks (having items $\Delta - \lfloor (\Delta - r)/2 \rfloor + 1, \ldots, \Delta$) are 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):

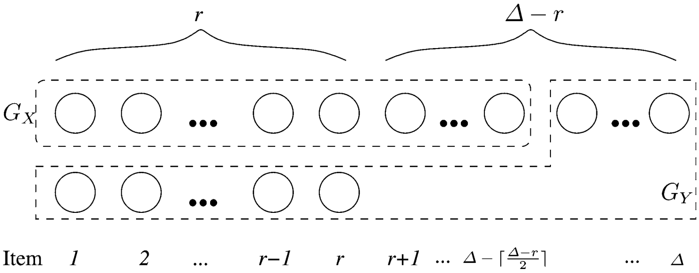

r
$\Delta - r$
G_X
G_Y
Item 1 2 ... r−1 r r+1 ... $\Delta - \lceil \frac{\Delta - r}{2} \rceil$ ... Δ

Fig. 8. The figure shows how the last group of disks in step 4 in algorithm Multi-Source Broadcast is partitioned into G_X and G_Y.

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

(a) The Gossip algorithm in [4] is applied to each group in parallel. After this step, each disk has all the items belonging to its group.

(b) In each round, disks in G_Y send item i to disks in G_X, where i is $\Delta - \lfloor(\Delta - r)/2\rfloor + 1, \ldots, \Delta$. 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 the reverse direction. Item i, where i is $r + 1, \ldots, \Delta - \lfloor(\Delta - r)/2\rfloor$, is sent from G_X to G_Y.

Thus, our algorithm takes $\lceil\log(N/\Delta)\rceil + 2\Delta$ rounds. The first term comes from the broadcast in step 1. The second term comes from the number of rounds required by the gossiping algorithm.

4.2. Analysis

Lemma 4.1. *For a group of disks of size $\Delta + r$, where $1 \leqslant r < \Delta$, if every disk has one item, exactly two disks have items $1, \ldots r$, and exactly one 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: step 4(a) can be done in $2(\Delta - (\Delta - r)/2)$ rounds because $\Delta - (\Delta - r)/2$ is the group size. In steps 4(b) and 4(c), 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 $(\Delta - r)/2 + (\Delta - r - (\Delta - r)/2)$ items to be sent in these 2 steps. Thus, the total number of rounds needed is

$$2\left(\Delta - \frac{\Delta - r}{2}\right) + \left(\frac{\Delta - r}{2}\right) + \left(\Delta - r - \frac{\Delta - r}{2}\right) = 2\Delta.$$

Case II. If $\Delta + r$ is odd and $|G_X| = \Delta - (\Delta - r - 1)/2$ is even: step 4(a) can be done in $2(\Delta - (\Delta - r - 1)/2 - 1)$ rounds. In step 4(b), $(\Delta - r - 1)/2$ items have 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 an item in each round. After this step finishes, only $(\Delta - r - 1)/2$ disks in G_X miss an item, while other disks in G_X receive all $(\Delta - r - 1)/2$ items. By using one more round, all disks in G_X can receive all items needed from G_Y. In step 4(c), $\Delta - r - (\Delta - r - 1)/2$ items have to be copied to G_Y, and we have enough source disks in G_X. Thus, it requires

$$2\left(\Delta - \frac{\Delta - r - 1}{2} - 1\right) + \left(\frac{\Delta - r - 1}{2} + 1\right) + \left(\Delta - r - \frac{\Delta - r - 1}{2}\right) = 2\Delta$$

rounds.

Case III. If $\Delta + r$ is odd and $|G_X| = \Delta - (\Delta - r - 1)/2$ is odd: Since $|G_X|$ is odd, step 4(a) takes $2(\Delta - (\Delta - r - 1)/2)$ rounds. We claim that in this step, in addition to receiving items from its group, all disks in G_X, except the disk that has item 1 originally,

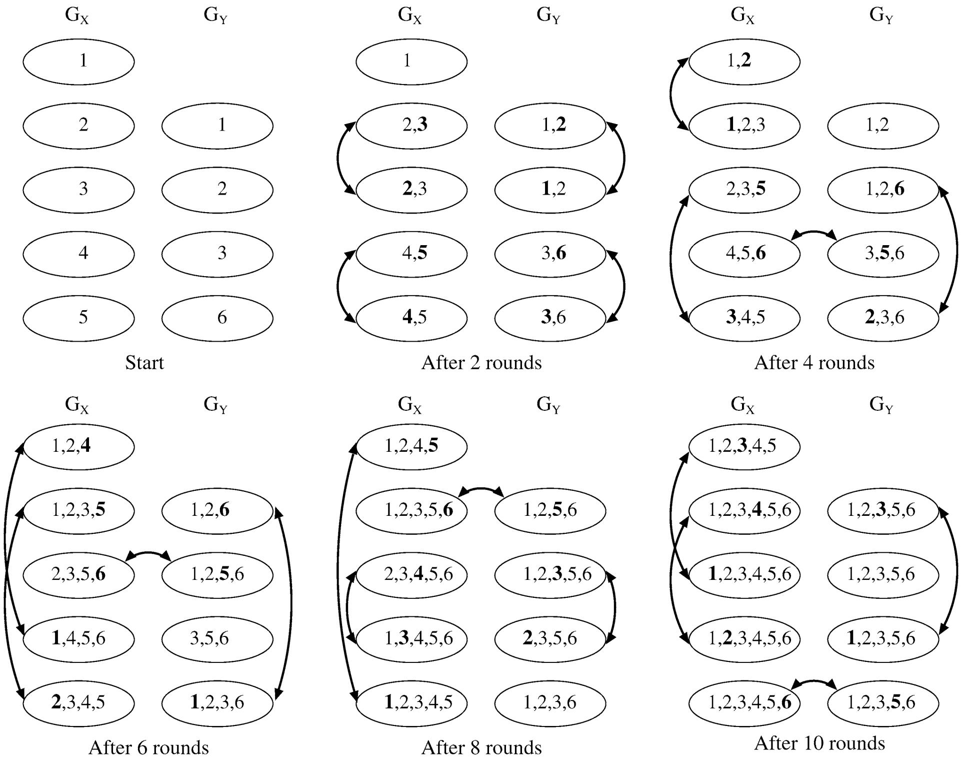


Fig. 9. An example of Case III in Section 4, with $\Delta = 6$ and $r = 3$. Recently received items are in bold.

have item Δ, and all disks in G_Y have item $\Delta - (\Delta - r - 1)/2$ (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 that has item 1 originally should be idle during the first two rounds, and (ii) the disk that received item $\Delta - (\Delta - r - 1)/2$, 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. The disk that has item $\Delta - (\Delta - r - 1)/2$ originally is idle during the last 2 rounds. We sort the disks in G_X according to the item number it has, and label the disks as disk $1, 2, \ldots, \Delta - (\Delta - r - 1)/2$. We also sort disks in G_Y, but label the disks as $2, 3, \ldots, \Delta - (\Delta - r - 1)/2$. 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 $\Delta - (\Delta - r - 1)/2$, 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 Fig. 9. In this example, $\Delta = 6$ and $r = 3$. Hence G_X has 5 items and G_Y has 4 items. Performing a gossip operation in G_X and G_Y takes 10 rounds (twice the size of the larger group). Notice that in 10 rounds, we are also able to send the largest item (item 6) in G_Y to all the disks in G_X except disk 1. At the same time, we are able to send the largest item (item 5) in G_X

to all the disks in G_Y. Thus when we send items between G_X and G_Y we are able to save rounds.

Note that we just exploit the idle cycles in the gossiping schedule. The number of rounds required is still $2(\Delta - (\Delta - r - 1)/2)$. 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 the other group.

In steps 4(b) and 4(c), 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 the other group in step 4(a). The disk in G_X, which does not have item Δ, can receive it in the last round of step 4(b) because $(\Delta - r - 1)/2 + 1 \leqslant |G_Y|$.

Thus, the total number of rounds is

$$2\left(\Delta - \frac{\Delta - r - 1}{2}\right) + \left(\frac{\Delta - r - 1}{2}\right) + \left(\Delta - r - \frac{\Delta - r - 1}{2} - 1\right) = 2\Delta. \qquad \square$$

The proof of the following theorem follows trivially from the above lemma.

Theorem 4.2. *Our Multi-Source Broadcast algorithm takes* $\lceil \log(N/\Delta) \rceil + 2\Delta$ *rounds.*

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

Theorem 4.3. *The time required for any migration instance of multi-source broadcasting is at least* $\lfloor \log(N/\Delta) \rfloor + 2(\Delta - 1)$.

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

In the initial $\lfloor \log(N/\Delta) \rfloor$ 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 $\lfloor \log(N/\Delta) \rfloor$ rounds. Now, at most $N - \Delta$ transfers are done.

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

$$\left\lfloor \log \frac{N}{\Delta} \right\rfloor + \frac{2N(\Delta - 1)}{N} = \left\lfloor \log \frac{N}{\Delta} \right\rfloor + 2(\Delta - 1). \qquad \square$$

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

5. Multi-source multicasting

We assume that we have N disks. Disk i, $1 \leqslant i \leqslant \Delta \leqslant N$, 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 disks. Recall that bypass disks are disks that are used as temporary holding points of data.

We define β as $\max_{j=1,\dots,N} |\{i \mid j \in D_i\}|$. 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.

We first present a simple 4-approximation for this problem. We then show how to improve it to a $(3 + o(1))$-approximation. After that we present a 3-approximation using bypass nodes.

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 as its source, is also the destination for several other data items.

Algorithm Multi-Source Multicast

1. We first compute a disjoint collection of subsets of disks G_i, $i = 1, \dots, \Delta$. We ensure that $G_i \subseteq D_i$ and $|G_i| = \lfloor |D_i|/\beta \rfloor$. (In Lemma 5.1, we will show how such G_i's can be obtained by using network flows.)
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 each node in G'_i sends item i to at most $\beta - 1$ disks and each node in $D_i \setminus G_i$ receives item i from one disk in G'_i. (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 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 $i = 1, \ldots, \Delta$, such that $|G_i| = \lfloor |D_i|/\beta \rfloor$ and $G_i \subseteq D_i$.*

The proof was shown as Lemma 3.2 in [19]. We include it here for completeness.

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

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

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 $\lfloor |D_i|/\beta \rfloor$. 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 $\lambda_i = \lfloor |D_i|/\beta \rfloor$. Send $1/\beta$ units of flow along $\lambda_i \beta$ outgoing edges from i. Note that since $\lambda_i \beta \leqslant |D_i|$ 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 $1/\beta$ 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. □

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 $\beta \geqslant 2$ 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 makes odd cycles (we will consider the case of odd cycles later).

Because the sets G_i are disjoint, it 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 $\beta \geqslant 2$, $\lceil \log |G_i| \rceil \leqslant \lceil \log(|D_i|/\beta) \rceil \leqslant \lceil \log |D_i| - 1 \rceil$.

Now let us consider the case of odd cycles. If any 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's is one, then this step can be done in 3 rounds (no broadcasting is needed inside G_i) and therefore the lemma is true. □

Lemma 5.3. *Consider a transfer graph only for item i in step* 3. *We can construct a transfer graph for item i 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, to $D_i \setminus G_i$ is at most $\beta - 1$.*

Proof. We divide each $D_i \setminus G_i$ into disjoint sets $D_{i1}, \ldots, D_{im_i}$ where $m_i = \lceil |D_i|/\beta \rceil$ such that $|D_{ij}| = \beta - 1$ for $j = 1, \ldots, m_i - 1$ 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 $|\{i \mid j \in D_i\}|$, i.e., the number of different sets D_i, that disk j belongs to. Thus, the in-degree of disk j in any solution (not using bypass disks) is β_j. Consider any source disk s_i for item i. In the transfer graph 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, $OPT \geqslant \max_i (\beta_{s_i} + 1)$. The maximum degree of any source disk s_i in the transfer graph is $\max_i \beta_{s_i} + (\beta - 1) + (\beta - 1) \leqslant OPT + 2\beta - 3$. Consider any disk j which is not the source, its total degree is $\beta_j + (\beta - 1)$. Note that $OPT \geqslant \max_j \beta_j$ and $\beta \geqslant 2$, the maximum degree of any non-source disk is $\max_{j \neq s_i} \beta_j + (\beta - 1) = OPT + (\beta - 1) \leqslant OPT + 2\beta - 3$. 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 a 3-approximation for this special case.

5.1. $(3 + o(1))$-approximation algorithm

In this section we present a polynomial time approximation algorithm with a factor of $3 + o(1)$ for the multi-source multicast problem.

In the previous algorithm, the sets G_i were disjoint. 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 disk 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 for which disk 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 ($\leqslant \beta$) different G_i sets. In other words, imagine that there are p slots in each disk, and each G_i will occupy exactly $\lfloor p|D_i|/\beta \rfloor$ slots. If G_i occupies a slot in a disk, the disk will be responsible for sending the item to either $\lfloor \beta/p \rfloor - 1$ or $\lceil \beta/p \rceil - 1$ 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 $\lfloor p|D_i|/\beta \rfloor$. In addition, each disk belongs to at most p subsets. 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 disk j, we would like the disk to satisfy either $\lfloor \beta/p \rfloor - 1$ or $\lceil \beta/p \rceil - 1$ 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 $\lfloor \lfloor p|D_i|/\beta \rfloor (\beta/p - 1) \rfloor$ 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 $\lceil \beta/p \rceil$ disks.

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

Proof. The basic idea of the proof is similar to that of Lemma 5.1.

First note that we have enough slots for G_i (we have N disks and each disk has p slots).

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

Now we show how to assign G_i to the slots using a flow network. 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 $\lambda_i = \lfloor p|D_i|/\beta \rfloor$. 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 $\lambda_i/|D_i|$ units of flow along each of the $|D_i|$ outgoing edges from i. Note that since $\lambda_i/|D_i| \leqslant p/\beta \leqslant 1$ 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 $\lambda_i/|D_i| \leqslant p/\beta$ 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 $\lfloor p|D_i|/\beta \rfloor$. □

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

Proof. Observe that sending items from disk 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 $(\max_i \log\lfloor p|D_i|/\beta \rfloor + 2) + (p + ((p-1) + (p-1))) + 4 = \max_i \log\lfloor p|D_i|/\beta \rfloor + 3p + 4$ rounds (by Theorem 5.5). □

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 $(\beta - p) + \lceil \beta/p \rceil$, 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 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 $\gamma_i = \lfloor \lfloor p|D_i|/\beta \rfloor (\beta/p - 1) \rfloor$. We add edges from node $i \in U$ to $j \in W$ if $j \in G_i$ with capacity $\lceil \beta/p \rceil - 1$. 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 $\gamma_i / \lfloor p|D_i|/\beta \rfloor$ units of flow along each of the $|G_i|$ outgoing edges from i. It is easy to see that $\gamma_i / \lfloor p|D_i|/\beta \rfloor \leqslant \beta/p - 1$, 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 $\beta/p - 1$ 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, $\lfloor p|D_i|/\beta \rfloor$ disks are satisfied in step 2 since they belong to G_i, and G_i can satisfy $\lfloor \lfloor p|D_i|/\beta \rfloor (\beta/p - 1) \rfloor$ disks in step 3. The number of disks that still need item i are:

$$\begin{aligned}&|D_i| - \left\lfloor p\frac{|D_i|}{\beta}\right\rfloor - \left\lfloor \left\lfloor p\frac{|D_i|}{\beta}\right\rfloor \left(\frac{\beta}{p} - 1\right)\right\rfloor \\ &\quad = |D_i| - \left\lfloor \left\lfloor p\frac{|D_i|}{\beta}\right\rfloor \frac{\beta}{p}\right\rfloor \leqslant |D_i| - \left\lfloor |D_i| - \left\lceil \frac{\beta}{p}\right\rceil\right\rfloor = \left\lceil \frac{\beta}{p}\right\rceil.\end{aligned}$$

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

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* $\max_i \log\lfloor p|D_i|/\beta \rfloor + 2\beta + \lceil \beta/p \rceil + 4p + 6$. *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 $2\beta + \lceil \beta/p \rceil + p + 2$ 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 disks

The main idea is that without bypass disks, only a small fraction of the N disks are 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 disks then we do not require that G_i is a subset of D_i. With bigger G_i sets, we can reduce the out-degree of the transfer graphs and thus reduces the total number of rounds.

Algorithm Multi-Source Multicast allowing bypass disks

1. We define $\bar{\beta}$ as $\frac{1}{N}\sum_{i=1,\ldots,N} |\{j \mid i \in D_j\}|$. In other words, $\bar{\beta}$ is the average number of items a disk requires, averaging over all disks. We arbitrarily choose a disjoint collection of subsets G_i, $i = 1, \ldots, \Delta$, with a constraint that $|G_i| = \lfloor |D_i| / \lceil \bar{\beta} \rceil \rfloor$. By allowing bypass disks, 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 $\lceil \bar{\beta} \rceil$ edges from each disk in G_i to satisfy $\lceil \bar{\beta} \rceil \cdot \lfloor |D_i| / \lceil \bar{\beta} \rceil \rfloor$ disks in D_i, and add at most another $\lceil \bar{\beta} \rceil - 1$ edges from disk i to satisfy the remaining disks in D_i.
4. This is the same as step 4 of the Multi-Source Multicast algorithm.

Theorem 5.12. *The total number of rounds required for the Multi-Source Multicast algorithm, by allowing bypass disks, is* $\max_i \lceil \log |D_i| \rceil + \beta + \lceil 2\bar{\beta} \rceil + 6$.

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

$$\sum_i |G_i| \leqslant \sum_i \frac{|D_i|}{\lceil \bar{\beta} \rceil} \leqslant \frac{1}{\bar{\beta}} \sum_i |D_i|.$$

Note that $\sum_i |D_i|$ is $\bar{\beta} N$ by the definition of $\bar{\beta}$. 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 $\lceil \bar{\beta} \rceil$ 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 $\lceil \bar{\beta} \rceil + (\lceil \bar{\beta} \rceil - 1)$. The multiplicity of the graph is still at most 4. Thus, the total number of rounds is $(\max_i \lceil \log |D_i| \rceil + 2) + \beta + \lceil \bar{\beta} \rceil + (\lceil \bar{\beta} \rceil - 1) + 4 \leqslant \max_i \lceil \log |D_i| \rceil + \beta + \lceil 2\bar{\beta} \rceil + 6$. □

We now argue that $\lceil 2\bar{\beta} \rceil$ is a lower bound on the optimal number of rounds. Intuitively, on average, every disk has to spend $\bar{\beta}$ rounds to send data, and another $\bar{\beta}$ rounds to receive data. As a result, the total number of rounds cannot be smaller than $\lceil 2\bar{\beta} \rceil$. 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 disks 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.

Appendix A. NP-hardness

We will prove the multi-source multicasting problem to be NP-hard by showing a reduction from a restricted version of 3SAT. Papadimitriou [23] 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 appears at least once in the given instance. If not, we can always simplify the instance so that each literal appears at least once.

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 $\bar{x}_i$, (iv) a set of destination disks $\overline{X}_i$ of size 3 which need item $\bar{x}_i$, (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, $\bar{x}_i$ 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 (the size should be 4 instead, if there are only two literals in clause j) that need item c_j. Moreover, for each literal in clause j, arbitrarily pick one disk in the set of destination disks corresponding to the literal, and that disk, which originally only needs the item corresponding to the literal, will also need item c_j. For example, if clause j is $x_p \vee \bar{x}_q \vee x_r$, 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 $\bar{x}_i$ 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, two disks in X_i received item x_i, while only 1 disk in $\overline{X}_i$ received item $\bar{x}_i$. 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 two disks in X_i are idle and they can receive item c_j from other disks. Furthermore, since a disk in X_i gets item x_i in round 1 (but not in round 2), it is not difficult to see that the two disks in X_i can receive item c_j from other disks in either round 2 or round 3, and still all requests in X_i can be satisfied. On the other hand, the remaining two 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 $\bar{x}_i$ 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 $\bar{x}_i$ in round 2. As a result, two disks in $\overline{X}_i$ are idle in round 3, while all disks in X_i are busy in round 3.

We claim that both disks in C_j, for all j, can be satisfied as well. For example, if clause j is $x_p \vee \bar{x}_q \vee x_r$, and suppose x_p is *true* in a satisfying assignment. From the argument above, there exists a schedule such that the disk in X_p, which needs x_p and c_j, is idle in round 3. However, if $\bar{x}_q$ and x_r are *false*, the disk in $\overline{X}_q$, which needs $\bar{x}_q$ and c_j, and the disk in X_r, which needs x_r and c_j, are busy getting an item $\bar{x}_q$ and item x_r, respectively, in round 3. Even when $\bar{x}_q$ or x_r is *true*, we can schedule the transfers of item $\bar{x}_q$ and x_r such that the disk in $\overline{X}_q$, which needs $\bar{x}_q$ and c_j, or the disk in X_r, which needs x_r and c_j, is busy getting an item in round 3. We can do this because if variable x_i is *true*, two disks in X_i can receive item c_j at either round 2 or 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 two 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, namely the second disk in C_j and the disk in X_p. If there are only two literals in clause j, the argument is similar. We need to satisfy two more disks in C_j, but, in round 2, we need to satisfy only one disk, instead of two disks, which cannot contribute item c_j in round 3. Thus, all requested items that appeared in Part II are satisfied as well. □

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 that need item s_i, if we have to finish all transfers in 3 rounds, once a disk receives s_i, it will be busy until round 3. Note that all switch disks have to receive s_i, x_i and $\bar{x}_i$. Therefore, all switch disks have to receive item x_i and $\bar{x}_i$ in the first two rounds, and s_i in round 3. If switch disk i receives item x_i in round 1, we set literal $\bar{x}_i$ 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 clauses j and k. Two disks in X_i may 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, $x_p \vee \bar{x}_q \vee x_r$, a disk in X_p, a disk in $\overline{X}_q$, a disk in X_r, and both 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, clause j has at least one true literal. If there are only two literals in clause j, the argument is similar. Because C_j is larger, all requests of item c_j cannot be satisfied when both literals are *false*. □

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

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

References

[1] E. Anderson, J. Hall, J. Hartline, M. Hobbes, A. Karlin, J. Saia, R. Swaminathan, J. Wilkes, An experimental study of data migration algorithms, in: Proceedings of the Workshop on Algorithm Engineering, in: Lecture Notes in Comput. Sci., vol. 2141, Springer, New York, 2001, pp. 145–158.

[2] B. Baker, R. Shostak, Gossips and telephones, Discrete Math. 2 (1972) 191–193.

[3] J. Bermond, L. Gargano, S. Perennes, Optimal sequential gossiping by short messages, Discrete Appl. Math. 86 (1998) 145–155.

[4] J. Bermond, L. Gargano, A.A. Rescigno, U. Vaccaro, Fast gossiping by short messages, in: Proceedings of the International Colloquium on Automata, Languages and Programming, in: Lecture Notes in Comput. Sci., vol. 944, Springer, 1995, pp. 135–146.

[5] S. Berson, S. Ghandeharizadeh, R.R. Muntz, X. Ju, Staggered striping in multimedia information systems, ACM SIGMOD Record 23 (2) (1994) 79–90.

[6] J.A. Bondy, U.S.R. Murty, Graph Theory with Applications, Elsevier, New York, 1977.

[7] R.T. Bumby, A problem with telephones, SIAM J. Algebraic Discrete Methods 2 (1) (March 1981) 13–18.

[8] E.J. Cockayne, A.G. Thomason, Optimal multi-message broadcasting in complete graphs, Util. Math. 18 (1980) 181–199.

[9] G. De Marco, L. Gargano, U. Vaccaro, Concurrent multicast in weighted networks, in: Proceedings of the Scandinavian Workshop on Algorithmic Theory, Springer, 1998, pp. 193–204.

[10] A.M. Farley, Broadcast time in communication networks, SIAM J. Appl. Math. 39 (2) (1980) 385–390.

[11] P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks, Discrete Appl. Math. 53 (1994) 79–133.

[12] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella, A. Zhu, Approximation algorithms for data placement on parallel disks, in: Proceedings of the 11th Annual ACM–SIAM Symposium on Discrete Algorithms, ACM, 2000, pp. 223–232.

[13] J. Hall, J. Hartline, A. Karlin, J. Saia, J. Wilkes, On algorithms for efficient data migration, in: Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms, ACM, 2001, pp. 620–629.

[14] A. Hajnal, E.C. Milner, E. Szemeredi, A cure for the telephone disease, Canad. Math. Bull. 15 (3) (1972) 447–450.

[15] S.M. Hedetniemi, S.T. Hedetniemi, A. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1988) 129–134.

[16] J. Hromkovic, R. Klasing, B. Monien, R. Peine, Dissemination of information in interconnection networks (broadcasting and gossiping), in: D.-Z. Du, D.F. Hsu (Eds.), Combinatorial Network Theory, Kluwer Academic, the Netherlands, 1996, pp. 125–212.

[17] C.A.J. Hurkens, Spreading gossip efficiently, Nieuw Archief voor Wiskunde 5 (1) (2000) 208–210.

[18] S. Kashyap, S. Khuller, Algorithms for non-uniform size data placement on parallel disks, in: Proceedings of the 23rd Annual Conference on Foundations of Software Technology and Theoretical Computer Science, in: Lecture Notes in Comput. Sci., vol. 2914, Springer, New York, 2003, pp. 265–276.

[19] S. Khuller, Y.A. Kim, Y.C. Wan, Algorithms for data migration with cloning, SIAM J. Comput. 33 (2) (2004) 448–461.

[20] W. Knodel, New gossips and telephones, Discrete Math. 13 (1975) 95.

[21] H.M. Lee, G.J. Chang, Set to set broadcasting in communication networks, Discrete Appl. Math. 40 (1992) 411–421.

[22] D. Liben-Nowell, Gossip is synteny: incomplete gossip and an exact algorithm for syntenic distance, in: Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms, ACM, 2001, pp. 177–185.

[23] C.H. Papadimitriou, Computational Complexity, Addison–Wesley, 1994.

[24] D. Richards, A.L. Liestman, Generalizations of broadcasting and gossiping, Networks 18 (1988) 125–138.

[25] H. Shachnai, T. Tamir, On two class-constrained versions of the multiple knapsack problem, Algorithmica 29 (2001) 442–467.

[26] H. Shachnai, T. Tamir, Polynomial time approximation schemes for class-constrained packing problems, in: Approximation Algorithms for Combinatorial Optimization, in: Lecture Notes in Comput. Sci., vol. 1913, Springer, 2000, pp. 238–249.

[27] M. Stonebraker, A case for shared nothing, Database Engrg. 9 (1) (1986).

[28] R. Tijdeman, On a telephone problem, Nieuw Archief voor Wiskunde 19 (3) (1971) 188–192.

[29] V.G. Vizing, On an estimate of the chromatic class of a p-graph, Diskret. Analiz. 3 (1964) 25–30 (in Russian).