Broadcasting on Networks of Workstations*

Samir Khuller†
University of Maryland
College Park, MD 20742
samir@cs.umd.edu

Yoo-Ah Kim‡
University of Maryland
College Park, MD 20742
ykim@cs.umd.edu

Yung-Chun (Justin) Wan‡
University of Maryland
College Park, MD 20742
ycwan@cs.umd.edu

ABSTRACT

Broadcasting and multicasting are fundamental operations. In this work we develop algorithms for performing broadcast and multicast in clusters of workstations. In this model, sending a message from one machine to another machine in the same cluster takes 1 time unit, and sending a message to a machine in a different cluster takes C time units. Lowekamp and Beguelin proposed heuristics for this model, but their algorithms may produce broadcast times that are arbitrarily worse than optimal. We develop the first constant factor approximation algorithms for this model. Algorithm LCF (Largest Cluster First) for the basic model is simple, efficient and has a worst case approximation guarantee of 2. We then extend these models to more complex models where we remove the assumption that an unbounded amount of simultaneous communication may happen using the global network. The algorithms for these models build on the LCF method developed for the basic problem. Finally, we develop broadcasting algorithms for the postal model where the sending processor does not block for C time units when the message is in transit. Moreover, we assume that the messages are small and so the bottleneck is the latency.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.0 [**Discrete Mathematics**]: Applications

General Terms

Algorithms, Theory

*Research supported by NSF grants CCR-0113192 and CCF-0430650.

†Department of Computer Science and Institute for Advanced Computer Studies

‡Department of Computer Science

SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA.

Keywords

Broadcasting, Multicasting, Approximation Algorithms

1. INTRODUCTION

Networks of Workstations (NOWs) are a popular alternative to massively parallel machines and are widely used (for example the Condor project at Wisconsin [18] and the Berkeley NOW project [1]). By simply using off-the-shelf PC's, a very powerful workstation cluster can be created, and this can provide a high amount of parallelism at relatively low cost. With the recent interest in grid computing [9] there is an increased interest to harness the computing power of these clusters to have them work together to solve large applications that involve intensive computation. Several projects such as Magpie [15, 14] are developing platforms to allow applications to run smoothly by providing primitives for performing basic operations such as broadcast, multicast, scatter, reduce etc. Many of these primitives are implemented using simple heuristics. Our goal is to develop models, and an understanding of the difficult issues and challenges in implementing broadcast and multicast on such platforms. Several approximation algorithms have been developed in the literature, but they are for different models (typically an underlying communication graph exists that forbids any communication between non-adjacent nodes). The algorithms developed are computationally intensive and complex.

One fundamental operation that is used in such clusters, is that of *broadcast* (this is a primitive in many message passing systems such as MPI [17, 5, 10, 12]). Some of this framework has been extended to clustered wide area systems (see [15, 14]) of the type we are addressing. In addition it is used as a primitive in many parallel algorithms. The main objective of a broadcast operation is to quickly distribute data to the entire network for processing. Another situation is when the system is performing a parallel search, then the successful processor needs to inform all other processors that the search has concluded successfully. Various models for heterogenous environments have been proposed in the literature. One general model is the one proposed by Bar-Noy et al. [2] where the communication costs between links are not uniform. In addition, the sender may engage in another communication before the current one is complete. An approximation factor with a guarantee of $O(\log k)$ is given for the operation of performing a multicast. Other popular models in the literature generally assume an underlying communication graph, with the property that only nodes adjacent in this graph may communicate. Elkin and Kortsarz [7, 8] developed approximation algorithms on this model. They also show that it is NP-hard to approximate the problem within a ratio of $3 - \epsilon$. However, this model is too restrictive and allows direct communication only between nodes adjacent in a certain communication graph. Broadcasting efficiently is an essential operation and many works are devoted to this (see [19, 11, 13, 3, 4] and references therein).

Consider several clusters of workstations. Each local cluster (sometimes this is also called a subnet) is connected on a fast local area network, and inter-cluster communication is via a wide area network. In such situations, the time taken for a pair of machines in the same cluster to communicate, can be significantly smaller than the communication time of a pair of machines in different clusters. In fact, in the work by Lowekamp and Beguelin [16] they also suggest methods for obtaining the subnets/clusters based on communication delays between pairs of processors.

Motivated by this, the *communication model* we consider is the following. There are k clusters of machines. Cluster i has size n_i. We will assume that in one time unit, a machine can send a message to any one machine in its own cluster. However, sending a message from one machine to another machine in a different cluster takes C time units. Even if the machines in a cluster are heterogenous, their transmission times are usually much less than the communication time across clusters. We also assume that a machine can be sending or receiving a message from only one machine at any point of time (a *matching*). In this model Lowekamp and Beguelin [16] propose some simple heuristics for performing broadcast and multicast. However, these heuristics may produce solutions arbitrarily far from optimal.

One potential concern with the above model is that it allows an arbitrary number of processors to communicate simultaneously in every time step. This is of concern if the global network connecting the different clusters does not have enough capacity to permit such arbitrary communication patterns. There are several ways in which we can restrict the model. One model that we propose is the *bounded degree model* where each cluster i is associated with a parameter d_i that restricts the *number* of processors from this cluster that can communicate with processors outside this cluster in each time step. Another possible manner in which we may restrict global communication in each time step is to restrict the *total* number of simultaneous transfers that may be going on in each time step without restricting the number of transfers into/out of a single cluster. We call this model the *bounded size matching model*.

In addition, we consider a *postal model* [3] where each message simply has a latency of C time units when the message is sent from one processor to another processor belonging to a different cluster. The sender is busy for only one time unit while the message is being injected into the network. The message takes C units of transit time and the receiver is busy for one unit of time when the message arrives. This model essentially captures the communication pattern as discussed in several papers that deal with implementations of systems to support such primitives (see [15, 14]).

The results for these models can be described as follows. By α-approximation, we mean that the schedule we generate has the broadcast (multicast) time at most α times the optimal.

1. We develop algorithms for broadcasting and multicasting in the basic $1/C$ model, and show that these algorithms produce solutions where the time to perform the broadcast is not more than optimal by a factor of 2 (moreover, this bound is tight for both algorithms).

2. For the *bounded degree model* we show how to reduce the problem to an instance of the basic model to develop a factor 3 approximation. The corresponding bound for multicasting is 3.

3. For the *bounded size matching* model we develop an algorithm for which we can prove a factor 2 approximation. The corresponding bound for multicasting is 2.

4. For the *postal model* our algorithm has a bound of 3. In addition, we present another algorithm, called *Interleaved LCF* and show that the makespan is at most 2 times OPT' where OPT' is the minimum makespan among schedules that minimize the total number of global transfers.

In many of these cases the algorithm we develop, called Algorithm LCF (Largest Cluster First), plays a central role. The algorithm computes a broadcast schedule given a source node. However, once the cluster sizes are fixed, each cluster can compute a broadcast schedule B_i that is to be used when cluster i initiates the broadcast. When a cluster first receives a message, it simply needs to know which cluster initiated the broadcast, and can follow the schedule specified by B_i. We first show that there is a simple analysis that shows that the worst case broadcast time can be bounded by a factor of 3 for this algorithm. We then improve the *lower bound* by introducing the concept of "experiencing" inter-cluster transfers to improve the approximation factor to 2. This lower bound in a sense combines the difficulty of propagating messages to a large number of clusters with the fact that some of the clusters may be very large. However, we do not have a polynomial-time algorithm to find an optimal solution, or a proof that the problem is NP-hard.

The LogP model [6] suggests an alternative framework when dealing with machines in a single cluster. Broadcasting algorithms [13] for the LogP model have been developed and shown to be optimal. An interesting generalized model would be to have a "two level LogP" model with different parameters for the local networks (intra-cluster) and global networks (inter-cluster).

One issue with our broadcasting protocol is that it assumes knowledge of the sizes of the clusters. In some applications, the cluster sizes may not be known accurately in advance. For example, what effect can this have on the broadcasting algorithm Largest Cluster First? In the simplest model, we study the effect of having inaccurate information regarding the sizes of the clusters in Section 7. In fact, as we demonstrate, even if there is a multiplicative factor of 2 inaccuracy in the sizes of the clusters, there is hardly any change in the performance of the broadcast algorithms.

2. FORMAL DESCRIPTION OF PROBLEM

We assume we have k clusters of processors. Cluster K_i has size $n_i, i = 0 \ldots (k-1)$, the number of processors in the i^{th} cluster. The total number of processors, denoted by N, is $\sum_{i=0}^{k-1} n_i$. We will assume that the broadcast/multicast originates at a processor in K_0. We order the *remaining* clusters in non-increasing size order. Hence $n_1 \geq n_2 \geq \ldots \geq n_{k-1}$. Clearly, n_0 could be smaller or larger than n_1; since it is simply the cluster that originates the broadcast/multicast.

A message may be sent from a processor, once it has received the message. If the message is sent to a processor in its cluster, the message arrives one time unit later. If the message is sent to a processor in a different cluster then the message arrives C time units later. Both the sending and receiving processors are busy during those C time units. In addition, we assume that each cluster advertises a single address to which messages are sent. Each cluster thus receives a message only once at this machine and then the message is propagated to different machines in the cluster. Thus new machines may be added or dropped without having to inform other clusters of the exact set of new addresses (we only need to keep track of the sizes of the clusters). In some cases, the broadcast time can be reduced by having many messages arrive at the same cluster. *However, when we compare to the optimal solution we do not make any assumptions about the communication structure of the optimal solution.*

In Section 6 we consider a slightly different postal model where a processor is busy for only one time unit when it sends a message. The time a message arrives at a receiver depends on whether the sender and receiver are in the same cluster or not – it takes one time unit if it is a local transfer, and C time units otherwise.

3. BROADCASTING

The high level description of the algorithm is as follows. The source node first performs a local broadcast within its cluster. This takes $\lceil \log n_0 \rceil$ rounds. After all the nodes of K_0 have the message, we broadcast the message to the first n_0 clusters. Each node in K_0 sends a message to a distinct cluster. This takes exactly C rounds. Each cluster that receives a message, does a local broadcast within its cluster. All nodes that have received the message then send the message to distinct clusters. Again this takes C more rounds. While doing this, every node in K_0 keeps sending a message to a cluster that has not received a message as yet. Repeat this until all the processors receive the message. We call this algorithm *Largest Cluster First (LCF)* as we always choose the largest cluster as a receiver among clusters that have not received the message.

Algorithm *LCF*

1. Broadcast locally in K_0 (this takes $\lceil \log n_0 \rceil$ rounds).
2. Each cluster performs the following until all processors get informed.
   (a) cluster K_i in which all processors have messages, picks the first n_i clusters that have not received a message, and sends the message to them at every C time unit. Repeat until all clusters have at least one message.
   (b) Each cluster which received a message does local broadcasting until all processors in the cluster have messages[1].

[1]We interrupt all local broadcasting and do one global transfer, if the number of processors having the message is at least the number of clusters that have not received a message.

Note that in our algorithm each cluster receives only one message from other clusters. That is, the total number of global transfers is minimized (we need $k-1$ global transfers). This property is important since we want to avoid wasting wide area bandwidth which is expensive.

3.1 Analysis

In this subsection, we prove that LCF gives a 2-approximation. For the purpose of analysis, we modify LCF slightly. The makespan of the schedule by the *modified* algorithm may be worse than the original algorithm (but no better) and it is at most 2 times the optimal.

In *Modified LCF*, local and global phases take place in turn (see Figure 1). Let L_i be the set of clusters that receive the message at i-th global step. For example, L_0 includes K_0 and L_1 includes all clusters that receive the message from K_0 at the end of the first global phase. Let N_i be the total number of processors in clusters belonging to L_i. That is, $N_i = \sum_{K \in L_i} |K|$.

Algorithm *Modified LCF*

1. Broadcast locally in K_0. Then we have that $L_0 = K_0$ and $N_0 = n_0$.
2. At i-th step (repeat until all processors get informed)
   (a) Global phase: Pick $\sum_{j=0 \ldots i-1} N_j$ largest clusters that are not informed as yet. Each processor in $\bigcup_{j=0 \ldots i-1} L_j$ sends one message to each of those clusters.
   (b) Local phase: Clusters in L_i do local broadcasting.

Let p be the number of global transfer steps that *Modified LCF* uses.

THEOREM 3.1. *The broadcast time of our algorithm is at most* $2 \log N + pC + 3$.

Define A_i (B_i) to be the biggest (smallest) cluster in L_i. We need the following two lemmas to prove this theorem.

LEMMA 3.2. *For* $i = 0 \ldots p-1$, $n_0 \cdot |B_1| \cdots |B_i| \leq N_i$.

PROOF. We prove this by induction. For $i = 0$, it is true since $N_0 = n_0$. Suppose that for $i = k$ $(< p-1)$ we have $n_0 \cdot |B_1| \cdots |B_k| \leq N_k$. Since at $(k+1)$-th global transfer step, every node in N_k will send the message to a cluster in L_{k+1}, $|L_{k+1}| \geq N_k$. Furthermore, the size of clusters in L_{k+1} is at least $|B_{k+1}|$ by definition. Therefore,

$$\begin{aligned} N_{k+1} &= \sum_{K \in L_{k+1}} |K| \\ &\geq \sum_{K \in L_{k+1}} |B_{k+1}| \\ &= |L_{k+1}| \cdot |B_{k+1}| \\ &\geq N_k \cdot |B_{k+1}| \\ &\geq n_0 \cdot |B_1| \cdots |B_k| \cdot |B_{k+1}|. \end{aligned}$$

□

LEMMA 3.3. $\log |A_1| < \log N - (p-2)$.

PROOF. After we have all processors in A_1 receive the message we need $p-1$ more global transfer steps. With $|A_1|$ copies, we can make $|A_1| \cdot 2^i$ processors receive the message after i global transfer steps by doubling the number of copies in each global step. Therefore $|A_1| \cdot 2^{p-2} < N$ (otherwise, we do not need p-th global broadcasting step). $\square$

Proof of Theorem 3.1. The upper bound of the total broadcast time for local transfer phases is $\lceil \log n_0 \rceil + \lceil \log |A_1| \rceil + \cdots + \lceil \log |A_p| \rceil$. Since we have $|A_i| \leq |B_{i-1}|$ (for $2 \leq i \leq p$) in LCF, it is upper bounded by $\lceil \log n_0 \rceil + \lceil \log |A_1| \rceil + \lceil \log |B_1| \rceil + \cdots + \lceil \log |B_{p-1}| \rceil$. By Lemma 3.2 and Lemma 3.3, the total broadcast time only for local transfer steps is at most

$$\begin{aligned} & \lceil \log n_0 \rceil + \lceil \log |A_1| \rceil + \lceil \log |B_1| \rceil + \cdots + \lceil \log |B_{p-1}| \rceil \\ \leq \; & \log n_0 + \log |B_1| + \cdots + \log |B_{p-1}| + \log |A_1| + p + 1 \\ < \; & \log n_0 \cdot |B_1| \cdots |B_{p-1}| + \log N - (p-2) + p + 1 \\ \leq \; & \log N_{p-1} + \log N + 3 \\ \leq \; & 2 \log N + 3. \end{aligned}$$

Our schedule uses p global transfer steps, taking a total of pC additional rounds. $\square$

In the next lemma, we prove that pC is a lower bound on the optimal broadcast time. To prove this we count the number of inter-cluster transfers a processor *experiences* as follows. Given a broadcast schedule, let the path from the source to the processor i, be $a_0, a_1, \ldots, a_l = i$. That is, the source a_0 sends the message to a_1 and a_1 sends to a_2 and so on. Finally a_{l-1} sends the message to processor i $(= a_l)$. Let e_j $(j = 0 \ldots l-1)$ represent the number of processors that receive the message from a_j via inter-cluster transfers until a_{j+1} receives the message (including the transfer to a_{j+1} if they are in different clusters). In addition, let e_l be the number of processors that receive the message from processor i via inter-cluster transfers. That is, i sends the message to e_l processors in other clusters. Then we say processor i experiences e inter-cluster transfers where $e = \sum_{j=0}^{l} e_j$. Figure 4 shows an example of how to count the number of inter-cluster transfers that a processor experiences. In the example, processor i experiences 11 inter-cluster transfers. If there is any processor that experiences p inter-cluster transfers, then pC is a lower bound on the optimal solution.

LEMMA 3.4. *At least one node in the optimal solution experiences p inter-cluster transfers.*

PROOF. Imagine a (more powerful) model in which once a node in a cluster receives the message, all nodes in the cluster receives the message instantly (That is, local transfers take zero unit of time). In this model the broadcast time is given by the maximum number of inter-cluster transfers that any processor experiences. We will prove that LCF gives an optimal solution for this model. Since LCF uses p global transfer steps, the optimal broadcast time is pC in this model. Since this lower bound is for a stronger model, it also works in our model.

Suppose that there is a pair of clusters K_i and K_j $(0 < i < j \leq k)$ such that K_j receives the message earlier than K_i in the optimal solution. Let K_i receive the message at time t_i and K_j receive at time t_j $(t_i > t_j)$. Modify the solution as follows. At time $t_j - C$ we send the message to K_i instead of K_j and K_i performs broadcasting as K_j does until t_i (This can be done since the size of K_i is at least as big as K_j.) At time t_i, K_j receives the message and after that the same schedule can be done. This exchange does not increase the broadcast time and therefore, LCF gives an optimal solution for the model with zero local transfer costs.

We now prove the lemma by contradiction. Suppose that there is an optimal solution (for the original model) in which all processors experience at most $p-1$ inter-cluster transfers. Then in the model with zero local transfer costs we should be able to find a solution with broadcasting time $(p-1)C$ by ignoring local transfers, which is a contradiction. $\square$

Using the above lemma, we know that pC is a lower bound on the optimal broadcast time. Since $\log N$ is also a lower bound on the optimal solution, this gives us 3-approximation (and an additive term of 3). However, the lower bound pC considers only the global communications of the optimal solution. On the other hand, the lower bound $\log N$ only counts the local transfers. To get a better bound, we prove the following theorem that combines the lower bounds developed above.

THEOREM 3.5. *The optimal solution has to take at least $(p-1)(C-1) + \lceil \log \frac{N}{2} \rceil$ rounds.*

PROOF. Consider an optimal schedule, in the end all N nodes receive the message. We partition all nodes into two sets, S_l and S_s, where S_l contains all nodes which experienced at least $p-1$ inter-cluster transfers, and S_s contains all nodes which experienced at most $p-2$ inter-cluster transfers. We now show that $|S_s| < \frac{N}{2}$. Suppose this is not the case, it means that the optimal solution can satisfy at least $\frac{N}{2}$ nodes using at most $p-2$ inter-cluster transfers. Using one more round of transfers, we can double the number of nodes having the message and satisfy all N nodes. This is a contradiction, since we use less than p inter-cluster transfers. Therefore we have $|S_l| \geq \frac{N}{2}$. Since originally we have one copy of the message, satisfying nodes in S_l takes at least $\lceil \log \frac{N}{2} \rceil$ transfers. So at least one node in S_l experienced $\lceil \log \frac{N}{2} \rceil$ transfers (either inter-cluster or local transfers). We know that all nodes in S_l experienced at least $p-1$ inter-cluster transfers. The node needs at least $(p-1)C + (\lceil \log \frac{N}{2} \rceil - (p-1))$ rounds. $\square$

We now prove a central result about Algorithm LCF which will be used later.

THEOREM 3.6. *Our algorithm takes at most $2OPT + 7$ rounds. Moreover, it takes at most $2OPT$ rounds when both p and C are not very small (i.e., when $(p-2)(C-2) \geq 7$).*

PROOF. If C is less than 2, we can treat the nodes as one large cluster and do broadcasting. This takes at most $C\lceil \log N \rceil$ and is a 2-approximation algorithm. The problem is also trivial if p is 1, because in this case $n_0 \geq k$. Therefore we consider the case where both values are at least 2. Here we make use of Theorems 3.1 and 3.5.

$$\begin{aligned} & (2OPT + 7) - (2 \log N + pC + 3) \\ \geq \; & (2((p-1)(C-1) + \lceil \log \frac{N}{2} \rceil) + 7) - (2 \log N + pC + 3) \\ \geq \; & (p-2)(C-2) \\ \geq \; & 0 \qquad (\text{when } p \geq 2 \text{ and } C \geq 2). \end{aligned}$$

$\square$

3.2 Bad Example

There are instances for which the broadcast time of LCF is almost 2 times the optimal. Suppose that we have 2 clusters, K_0 and K_1, each of size n_0 and n_1 ($n_0 \leq n_1$), respectively. In addition, there are $n_0 - 1$ more clusters, each of size 1. A node in K_0 has a message to broadcast. It is easy to see that the makespan of our algorithm is $\lceil \log n_0 \rceil + C + \lceil \log n_1 \rceil$. However, the broadcasting can be made faster by sending a message to K_1 before local broadcast in K_0 is finished. A possible schedule is (i) make one local copy in K_0 (ii) one processor in K_0 send a message to K_1 and another processor does local broadcast in K_0 (iii) after finishing local broadcast, processors in K_0 send messages to the remaining $n_0 - 1$ clusters (iv) clusters other than K_0 do local broadcasting as soon as they receive a message. The makespan of this solution is $1 + \max\{C + \lceil \log n_1 \rceil, \lceil \log(n_0 - 1) \rceil + C\}$. In the case where $n_0 \approx n_1$ and $\log n_0 \gg C$, the makespan of LCF is almost 2 times the optimal.

4. MULTICASTING

For multicasting, we need to have only a subset of processors receive the message. We may reduce the multicast time significantly by making use of large clusters that may not belong to the multicast group. Let n'_i denote the number of processors in K_i that belong to the multicast group. Let M denote the set of clusters (except K_0) in which some processor wants to receive the message and k' denote the size of set M. Formally, $M = \{K_i | n'_i > 0 \text{ and } i > 0\}$ and $k' = |M|$.

Let $LCF(m)$ be algorithm LCF to make m copies. That is, $LCF(m)$ runs in the same way as LCF but stops as soon as the total number of processors that received the message is at least m (we may generate up to $2(m-1)$ copies). For example, LCF for broadcasting is $LCF(N)$. Here is the algorithm.

Algorithm *LCF Multicast*
Phase 1: Run $LCF(k')$ by using any processor whether it belongs to the multicast group or not.
Phase 2: Send one copy to each cluster in M if it has not received any message yet.
Phase 3: Do local broadcast in clusters of M.

4.1 Analysis

Let p' be the number of global transfer steps $LCF(k')$ uses. Suppose D be the number of rounds taken in the last local broadcast step in $LCF(k')$ (after the p'-th global transfer steps). Note that some nodes in clusters performing the last local broadcast may not receive the message, since we stop as soon as the total number of nodes having the message is at least k', and hence $D \leq \lceil \log |A_{p'}| \rceil$. Nevertheless, D may be greater than $\lceil \log |B_{p'}| \rceil$ and thus some clusters may stop local broadcast before the D-th round. Let $A_{p'+1}$ be the biggest cluster among clusters which have not received a copy after $LCF(k')$ has finished.

To get a 2-approximation algorithm, we need the following lemma, which bounds the sum of the number of rounds taken in the local phases in $LCF(k')$ and in the last local broadcast in clusters of M. The lemma still holds even when a cluster performing local broadcast in Phase 3 needs to broadcast to the whole cluster (i.e., $n'_i = n_i$).

LEMMA 4.1. $(\log n_0 \cdot |B_1| \cdots |B_{p'-2}| + D) + \max(\lceil \log |A_{p'}| \rceil - D, \lceil \log |A_{p'+1}| \rceil) \leq \log k' + 2$

PROOF. Case I: $\lceil \log |A_{p'}| \rceil - D > \lceil \log |A_{p'+1}| \rceil$. It is easy to see that $(\log n_0 \cdot |B_1| \cdots |B_{p'-2}| + D) + (\lceil \log |A_{p'}| \rceil - D) \leq \log n_0 \cdot |B_1| \cdots |B_{p'-1}| + 1 \leq \log k' + 1$, and the lemma follows.

Case IIa: $\lceil \log |A_{p'}| \rceil - D \leq \lceil \log |A_{p'+1}| \rceil$ and $D > \lceil \log |B_{p'}| \rceil$. Note that $2^D \leq 2|A_{p'}| \leq 2|B_{p'-1}|$ and $|A_{p'+1}| \leq |B_{p'}|$; we have $(\log n_0 \cdot |B_1| \cdots |B_{p'-2}| + D) + (\lceil \log |A_{p'+1}| \rceil) < \log n_0 \cdot |B_1| \cdots |B_{p'}| + 2 \leq \log N_{p'-1} \cdot |B_{p'}| + 2$. After the p'-th global transfer step, one node in each of $N_{p'-1}$ clusters has just received the message. Each of these clusters will generate at least $|B_{p'}|$ copies (since $D > \lceil \log |B_{p'}| \rceil$), so $N_{p'-1} \cdot |B_{p'}| < k'$, and the lemma follows.

Case IIb: $\lceil \log |A_{p'}| \rceil - D \leq \lceil \log |A_{p'+1}| \rceil$ and $D \leq \lceil \log |B_{p'}| \rceil$. Note that $|A_{p'+1}| \leq |B_{p'-1}|$; we have $(\log n_0 \cdot |B_1| \cdots |B_{p'-2}| + D) + (\lceil \log |A_{p'+1}| \rceil) \leq \log n_0 \cdot |B_1| \cdots |B_{p'-1}| \cdot 2^D + 1 \leq \log N_{p'-1} \cdot 2^D + 1$. After the p'-th global transfer step, each cluster which has just received the message will generate at least 2^D copies, so $\log N_{p'-1} \cdot 2^{D-1} < k'$, and the lemma follows. □

THEOREM 4.2. *Our multicast algorithm takes at most* $2 \log k' + p'C + C + 4$ *rounds.*

LEMMA 4.3. *At least one node in the optimal solution experiences* p' *inter-cluster transfers.*

THEOREM 4.4. *The optimal solution takes at least* $(p' - 1)(C - 1) + \lceil \log \frac{k'}{2} \rceil$ *rounds.*

THEOREM 4.5. *Our algorithm takes at most* $2OPT + 10$ *rounds. Moreover, it takes at most* $2OPT$ *rounds when both* p' *and* C *are not very small (i.e., when* $(p'-3)(C-2) \geq 10$*).*

PROOF. We can show that $(2OPT + 10) - (2 \log k' + p'C + C + 4) \geq (p' - 3)(C - 2)$. The problem is trivial when C is less than 2 or $p = 1$. When $p' = 2$, we can do an exhaustive search on the number of clusters in M which receives the message in the first global transfer step in $LCF(k')$. (details omitted). □

5. BOUNDING GLOBAL TRANSFERS

In the model we considered, we assume any node may communicate with any other node in other clusters, and the underlying network connecting clusters has unlimited capacity. A more practical model is to restrict the number of pairs of inter-cluster transfers that can happen simultaneously. In this section we present two models to restrict the network capacity. The *bounded degree model* restricts the number of inter-cluster transfers associated with a particular cluster, while the *bounded size matching model* restricts the total number of inter-cluster transfers at any given time.

5.1 Bounded Degree Model

Associate an additional parameter d_i with each cluster i, that limits the number of inter-cluster transfers from or to nodes in cluster i in a time unit. We call this limitation a degree constraint. We denote an instance of this model be $I(n_i, d_i)$, meaning that there are n_i nodes in cluster K_i, and at most d_i of those may participate in inter-cluster transfers at any given time.

Algorithm *Bounded Degree Broadcast*
Given Instance $I(n_i, d_i)$, arbitrarily select a subset K'_i of

d_i nodes in each cluster, and consider only the K_i'. We have a new instance $I(d_i, d_i)$. Note that $I(d_i, d_i)$ can be viewed as an instance of the general broadcast problem on the unrestricted model. In phase 1, run Algorithm LCF in Section 3 on $I(d_i, d_i)$. In phase 2, since there are d_i informed nodes in each cluster, we do local broadcasting to send the message to the remaining $n_i - d_i$ nodes.

An important observation is that since there is only a unique message, it does not matter which subset of nodes in a cluster perform inter-cluster transfers. What matters is the number of informed nodes in the clusters at any given time. The following lemma compares the optimal number of rounds taken by instances using the two different models.

LEMMA 5.1. *The optimal schedule of Instance $I(d_i, d_i)$ takes no more rounds than the optimal schedule of the corresponding instance $I(n_i, d_i)$.*

THEOREM 5.2. *Our algorithm takes at most $3OPT + 7$ rounds.*

5.2 Bounded Degree Model: Multicasting

In this model only a subset M_i (possibly empty) of nodes in cluster K_i needs the message. Nodes in $K_i \setminus M_i$ may help passing the message around. Let n_i' be $|M_i|$. Observe that although we may make use of nodes in $K_i \setminus M_i$, we never need more than d_i nodes in each cluster, because of the degree constraint in the number of inter-cluster transfers. Similarly if $d_i \leq n_i'$, nodes in $K_i \setminus M_i$ are never needed. Therefore set n_i to be $\max(d_i, n_i')$. Arbitrarily select d_i nodes for each cluster, with priority given to nodes in M_i. Run the LCF Multicast algorithm on the selected nodes. Now there are d_i nodes having the message in each cluster belongs to the multicast group, so we can do local broadcasting to satisfy the remaining nodes.

THEOREM 5.3. *Our algorithm takes at most $3OPT + 10$ rounds.*

5.3 Bounded Size Matching

In this model, we bound the number of inter-cluster transfers that can be performed simultaneously. Let us assume that we allow only B inter-cluster transfers at a time. Note that we can assume $B \leq \lfloor N/2 \rfloor$ since this is the maximum number of simultaneous transfers allowed by our matching-based communication model.

Algorithm *BoundedSizeBroadcast*
Phase 1: We run $LCF(B)$ to make B copies of the message.
Phase 2: Every C time units we make B more copies by inter-cluster transfers until all clusters have at least one copy of the message.
Phase 3: Do local broadcast to inform all the processors in each cluster.

Let p_B be the number of global transfer steps $LCF(B)$ uses, and p_L be the number of global transfer steps in the second stage of the algorithm.

THEOREM 5.4. *We need $2 \log B + p_B C + p_L C + 4$ rounds for broadcasting when we allow only B inter-cluster transfers at a time.*

LEMMA 5.5. *In any schedule, there is a processor that experiences $p_B + p_L$ inter-cluster transfers.*

THEOREM 5.6. *The optimal solution takes at least $(p_B + p_L - 1)(C - 1) + \lceil \log B \rceil$ rounds.*

THEOREM 5.7. *Our algorithm takes at most $2OPT + 6$ rounds.*

Remark: Note that by setting $B = \lfloor N/2 \rfloor$, we can improve the makespan of the basic broadcasting (without any bound on the global transfers) by one round. This is because in this algorithm we stop performing local transfers when the number of copies is $\lfloor N/2 \rfloor$ (as more copies cannot contribute to global transfers) and start global transfers.

5.4 Bounded Size Matching: Multicasting

In this model only a subset M_i of nodes in cluster K_i needs the message. Define $M = \{K_i | n_i' > 0 \text{ and } i > 0\}$ and $k' = |M|$. We assume $k' > B$ or otherwise we can use the *LCF Multicast* algorithm. We run $LCF(B)$ by using any processor available. Then every C time units we make B more copies by inter-cluster transfers until all clusters in M have at least one copy of the message. Lastly do local broadcast to inform all the processors in each cluster in M.

THEOREM 5.8. *The algorithm takes at most $2OPT + 10$ rounds.*

6. POSTAL MODEL

In this section, we consider a slightly different model. This model is motivated by the interest in Grid computing [9] and computing on clusters of machines. In fact, the work on the Magpie project [15, 14] specifically supports this communication model. In previous sections, we assumed that when processor p_i sends a message to processor p_j in another cluster, p_i is busy until p_j finishes receiving the message (this takes C time units). However, in some situations, it may not be realistic since p_i may become free after sending the message and does not have to wait until p_j receives the message.

In this section, we assume that a processor is busy for only one time unit when it sends a message. The time a message arrives at the receiver depends on whether the sender and receiver are in the same cluster or not—it takes one time unit if it is a local transfer (within the cluster), and C time units if it is an inter-cluster transfer.

We first show that LCF gives a 3-approximation in this model. In addition, we present another algorithm, which we call *Interleaved LCF*. Recall that we want to minimize the total number of global transfers as well as minimizing the makespan. Let OPT' denote the minimum makespan among all schedules that minimize the total number of global transfers. We can show that the makespan of the schedule generated by *Interleaved LCF* is at most 2 times OPT'.

6.1 Analysis of LCF

The analysis is similar to the one presented in Section 3.1. We modify the algorithm (for the analysis purpose) so that we have local and global phases in turn. However, in this model a processor can initiate more than one global transfer in a global phase since senders are busy for only one time unit per global transfer. We define A_i, B_i, L_i as follows. Let $A_0 = |K_0|$. After finishing local transfers in K_0, all processors in K_0 start global transfers. They can initiate a global transfer at every time unit. After C time units,

n_0 clusters receive a message (denoted as L_1). Let $A_1(B_1)$ be the biggest(smallest) cluster among them. Now K_0 stops global transfers for $\lceil \log A_1 \rceil$ rounds (global transfers already initiated continue to be done). For those $\lceil \log A_1 \rceil$ rounds, every cluster that received the message performs (only) local broadcasting. For all the clusters that receive the message in this step, they have exactly $\lceil \log A_1 \rceil$ rounds of local broadcasting. So for example, if a cluster receives t time unit later than A_1 then it can only start its global transfers t time units later than A_1 (it will be idle even if it finishes local broadcasting earlier). Note that A_1 is the biggest cluster in L_1 and therefore, $\lceil \log A_1 \rceil$ is enough for local broadcasting of those clusters. After $\lceil \log |A_1| \rceil$ time units, we have all clusters that have finished local broadcasting phase perform global transfers every time unit *for C rounds*. Clusters that have not finished local phase keep performing local broadcasting (or wait) and then start global transfers. Repeat this until all processors get informed. In general, we define L_i as clusters that receive messages in the first global transfers of i-th step (so they can participate in global phase of $(i+1)$-th step from the beginning). (See Fig 5(a)). A_i (B_i) is the biggest (smallest) cluster in L_i. Suppose that the schedule has p global phases. Then it is easy to see that Lemmas 3.2 and 3.3 hold. There is one subtle case where there are some clusters that receive the message later than A_p by the transfers initiated in p-th global phases (see Figure 5(b)).

THEOREM 6.1. *The makespan of* Modified LCF *is at most* $2 \log N + pC + 3$ *when* A_p *is one of the last clusters that receive the message.*

PROOF. The total makespan taken for local transfers is

$$\begin{aligned}
& \lceil \log n_0 \rceil + \lceil \log |A_1| \rceil + \cdots + \log \lceil |A_p| \rceil \\
\leq\; & \log n_0 + \log |A_1| + \cdots + \log |A_p| + (p+1) \\
\leq\; & \log n_0 + \log |A_1| + \log |B_1| \cdots + \log |B_{p-1}| + (p+1) \\
=\; & \log n_0 \cdot |A_1| \cdot |B_1| \cdots |B_{p-1}| + (p+1) \\
\leq\; & \log |A_1| + \log N_{p-1} + (p+1) \quad (\textit{by Lemma } 3.2) \\
\leq\; & 2 \log N + 3 \quad (\textit{by Lemma } 3.3)
\end{aligned}$$

Therefore, the makespan of the schedule is at most $2 \log N + pC + 3$. □

LEMMA 6.2. *If we assume that local transfers take zero unit of time, the makespan is at least pC when A_p is one of the last clusters that receive the message. If there are some clusters that receive the message c' time units later than A_p then the makespan is at least $pC + c'$.*

PROOF. Consider the schedule given by *LCF* when local transfers take zero units of time. It is easy to see that *LCF* gives an optimal solution in this case. Moreover it is the same as the schedule by the Modified *LCF* if we ignore local transfer phases. Since Modified *LCF* needs p global phases, pC is a lower bound for the optimal solution. □

We thus conclude:

THEOREM 6.3. *The makespan of the schedule generated by LCF is at most 3 times the optimal (with an additive term of 3) when A_p is one of the last clusters that receive the message.*

We now deal with the case when there are other clusters that receive the message later than A_p. Let A_{p+1} denote the biggest cluster that receives the message last, and it receives the message c' time units later than A_p ($c' < C$ since otherwise we would have another global phase). We need additional $c' + \lceil \log |A_{p+1}| \rceil$ time units (it is less than $c' + \lceil \log |B_p| \rceil$).

LEMMA 6.4. $n_0 \cdot |B_1| \cdots |B_p| \leq N_p$.

LEMMA 6.5. *$pC + c'$ is a lower bound on the optimal solution.*

PROOF. Suppose that local transfers take zero unit of time. Then in *LCF*, A_{p+1} receives the message c' time units later than A_p (since the schedule can be obtained by ignoring local phases). Therefore $pC + c'$ is a lower bound on the optimal solution. □

THEOREM 6.6. *The makespan of the schedule generated by LCF is at most 3 times the optimal (with additive term of 4).*

6.2 Interleaved LCF

We present another algorithm called *Interleaved LCF*. We show that it is 2-approximation among schedules that use the minimum number of global transfers.

Algorithm *Interleaved LCF*

At every two rounds, a processor that has the message alternately performs the following two steps.

1. Local transfer: if there is any processor in the same cluster that has not received the message, then send the message to it.

2. Global transfer: if there is any cluster that has not received the message, choose the biggest cluster among them and send the message to a processor in the cluster.

We only consider a set of schedules (denoted as S) that minimize the total number of global transfers. Note that schedules in S have the property that each cluster receives only one message from outside ($k-1$ in total). Let OPT_S be the minimum makespan among all schedules in S.

LEMMA 6.7. *There is a schedule in S with makespan OPT_S in which for any pair of clusters K_i, K_j ($n_i > n_j$), K_i receives a message no later than K_j.*

PROOF. Given a schedule in S with makespan OPT_S, if there is a pair of clusters K_i, K_j ($n_i > n_j$) and K_i receives a message at time t_i and K_j receives a message at time t_j ($t_i > t_j$), then we can modify the schedule so that K_i receives the message no later than K_j without increasing the makespan.

At t_j K_i (instead of K_j) receives the message. K_i can do all transfers that K_j does till time t_i. At time t_i, K_j receives a message. Let x_t processors in K_i received the message just after time t in the *original* schedule. Similarly, let y_t processors in K_j received the message just after time t. Then $x_{t_i} = 1$ and $y_{t_i} \leq n_j$. Note that we cannot swap the roles of two clusters just after t_i since K_i has y_{t_i} messages and K_j has only one message. Therefore, K_i should keep performing transfers as if it is K_j for some time. Let t' be the last time when $x_{t'} \leq y_{t'}$. That is, $x_{t'+1} > y_{t'+1}$.

At time $t'+1$ we need to carefully choose which transfers we should do. Note that just before time $t'+1$, K_i has $y_{t'}$ messages and K_j has $x_{t'}$ messages. In K_i we choose $x_{t'+1} - y_{t'}$ processors to make local transfers so that after $t'+1$, K_i has $x_{t'+1}$ copies of message. Since $x_{t'+1} \leq 2x_{t'} \leq 2y_{t'}$, $x_{t'+1} - y_{t'} \leq y_{t'}$ and therefore, we have enough processors to choose. Similarly, in K_j $y_{t'+1} - x_{t'}$ processors do local transfers so that K_j has $y_{t'+1}$ after $t'+1$. The total number of global transfers coming from K_i and K_j in the original schedule is at most $x_{t'} + y_{t'} - (x_{t'+1} - x_{t'}) - (y_{t'+1} - y_{t'}) = x_{t'} + y_{t'} - (y_{t'+1} - x_{t'}) - (x_{t'+1} - y_{t'})$ and this is exactly the number of remaining processors. Therefore, we have the remaining processors enough to make global transfers. After $t'+1$, we can do transfers as in the original schedule. $\square$

We can now consider schedules with the property in Lemma 6.7 only. Due to the property, a processor knows the receiver to send a message when it performs a global transfer— the largest cluster that has not received any message. The only thing a processor needs to decide at each time is whether it will make a local transfer or global transfer. By performing local and global transfer alternatively, we can bound the makespan by a factor of two.

THEOREM 6.8. *The makespan of* Interleaved LCF *is at most 2 times OPT_S.*

PROOF. Given an optimal schedule with the property in Lemma 6.7, modify the schedule so that each operation takes 2 units of time. That is, if a processor performs a local transfer then it is idle in the next time slot and if a processor performs a global transfer, it is idle in the previous time slot. The makespan of the modified schedule is at most 2 times the optimal. It is easy to see that the schedule by *Interleaved LCF* should not be worse than the modified schedule since in *Interleaved LCF*, the processors performs local and global transfers alternatively with no idle time. $\square$

7. EXPERIMENTS

One issue with our broadcasting protocol is that it assumes knowledge of the sizes of the clusters. In some applications, the cluster sizes may not be known accurately in advance. What effect can this have on the broadcasting algorithm Largest Cluster First for example? In the simplest model, we study the effect of having inaccurate information regarding the sizes of the clusters.

We run the LCF algorithm using the correct cluster sizes; in addition, we run the LCF algorithm by basing the order on advertised sizes that may be off by a factor of 2. For example, for each cluster we let the advertised size be fixed, but change the actual size randomly by either doubling it, or halving it. We now run the protocol where the cluster ordering is made by using the advertised size. We found that there is hardly any change in the performance of the broadcasting algorithm.

In the following experiment, we have roughly 2000 clusters and each cluster has advertised size between 1 and 100 nodes. We choose the actual cluster size using a Zipf distribution, i.e., Prob(a cluster having size i) $= \frac{c}{i^{1-\theta}}$, $\forall i = 1, \ldots, 100$ and $0 \leq \theta \leq 1$, where $c = \frac{1}{H_M^{1-\theta}}$, $H_M^{1-\theta} = \sum_{j=1}^{100} \frac{1}{j^{1-\theta}}$, and θ determines the degree of skewness. We assign θ to be 0. Figure 2 shows a histogram of the cluster sizes. We also let C be a parameter and vary this from 10 to 1000. This is a reasonable range as we expect the time to send a message across clusters to be within this range, assuming that the time to send a message within a cluster takes unit time. (If it takes a few milliseconds to send a message from one machine to another locally, it may take upto a few hundred, or thousand milliseconds to send a message to a machine belonging to another cluster.)

(*Actual*) records the broadcast time when the cluster sizes are the same as their advertised sizes. (*Advertised*) records the broadcast time when the cluster sizes are either half or double their advertised sizes (this choice is made independently and randomly for each cluster). We also compare both these methods to the broadcast time if the algorithm were to use a random ordering of the clusters (*Random*). We also illustrate these times and compare them to the best lower bounds from Section 3.1.

7.1 Results

As shown in Figure 3, we found that (*Actual*) performs the best. This is unsurprising because it has completely accurate information. Note that in all the experiments we ran, it performs at most 1.5 times the lower bound, which is smaller than the theoretical bound of 2. (However, we believe that the broadcasting time is a much closer to the optimal solution even though our lower bounds are not strong enough to argue this.) Moreover, (*Advertised*) performs very close to (*Actual*) (it takes only one more round in all instances). This behavior shows that one needs not to have completely accurate information on the size of the clusters for our Largest Cluster First algorithm to perform well. On the other hand, if the algorithm uses a random ordering of the clusters, it takes, on average, at least 24% more number of rounds than (*Actual*). In addition, as C increases, (*Actual*) performs closer and closer to the optimal.

8. REFERENCES

[1] T. Anderson, D. Culler, D. Patterson, and the NOW team. A case for NOW (networks of workstations). *IEEE Micro*, 15(1):54–64, Feb. 1995.

[2] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Multicasting in heterogeneous networks. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 448–453, 1998.

[3] A. Bar-Noy and S. Kipnis. Designing broadcast algorithms in the postal model for message-passing systems. *Mathematical Systems Theory*, 27(5), 1994.

[4] P. Bhat, C. Raghavendra, and V. Prasanna. Efficient collective communication in distributed heterogeneous systems. *Journal of Parallel and Distributed Computing*, 63(3):251–263, 2003.

[5] J. Bruck, D. Dolev, C. Ho, M. Rosu, and R. Strong. Efficient message passing interface (MPI) for parallel computing on clusters of workstations. *Journal of Parallel and Distributed Computing*, 40(1):19–34, Jan. 1997.

[6] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 1–12, 1993.

[7] M. Elkin and G. Kortsarz. Combinatorial logarithmic approximation algorithm for the directed telephone

broadcast problem. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 438–447, 2002.

[8] M. Elkin and G. Kortsarz. A sublogarithmic approximation algorithm for the undirected telephone broadcast problem: a path out of a jungle. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 76–85, 2003.

[9] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[10] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[11] S.M. Hedetniemi, S.T. Hedetniemi, and A.L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1988.

[12] P. Husbands and J. Hoe. MPI-StarT: Delivering network performance to numerical applications. In *Proceedings of the IEEE/ACM SC98 Conference*, page 17, Nov. 1998.

[13] R. Karp, A. Sahay, E. Santos, and K. Schauser. Optimal broadcast and summation in the LogP model. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 142–153, 1993.

[14] T. Kielmann, H. Bal, and S. Gorlatch. Bandwidth-efficient collective communication for clustered wide area systems. In *Proceedings of the 14th International Symposium on Parallel and Distributed Processing*, page 492, 2000.

[15] T. Kielmann, R. Hofman, H. Bal, A. Plaat, and R. Bhoedjang. MagPIe: MPI's collective communication operations for clustered wide area systems. In *Proceedings of the 7th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 131–140, 1999.

[16] B. Lowekamp and A. Beguelin. ECO: Efficient collective operations for communication on heterogeneous networks. In *Proceedings of the 10th International Parallel Processing Symposium*, pages 399–405, Honolulu, HI, Apr. 1996.

[17] Message passing interface forum, Mar. 1994.

[18] J. Pruyne and M. Livny. Interfacing condor and PVM to harness the cycles of workstation clusters. *Journal on Future Generations of Computer Systems*, 12(1):67–85, 1996.

[19] D. Richards and A. L. Liestman. Generalization of broadcasting and gossiping. *Networks*, 18:125–138, 1988.

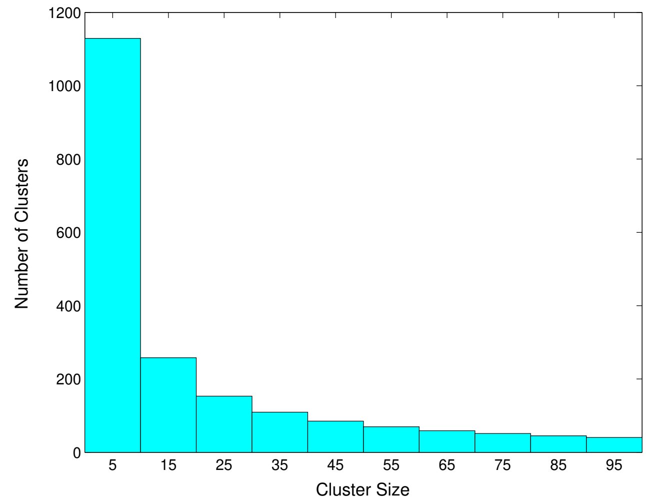

Figure 2: A histogram of the cluster sizes.

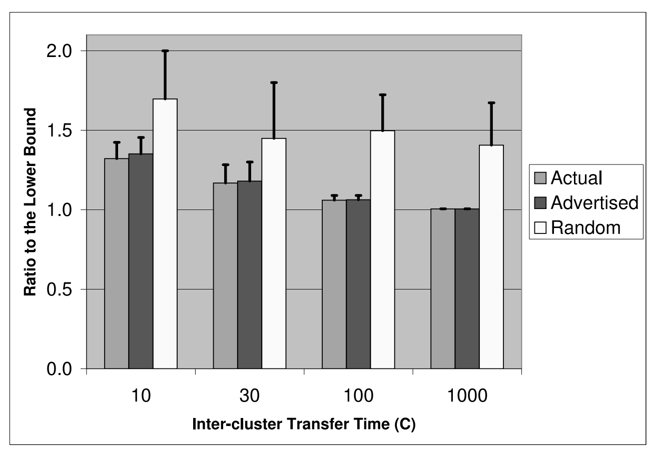

Figure 3: The ratio of the number of rounds taken by the algorithms to the lower bound, averaged over 5 inputs, with different values of C (10, 30, 100, and 1000). The maximum ratio appears on the top of each bar.

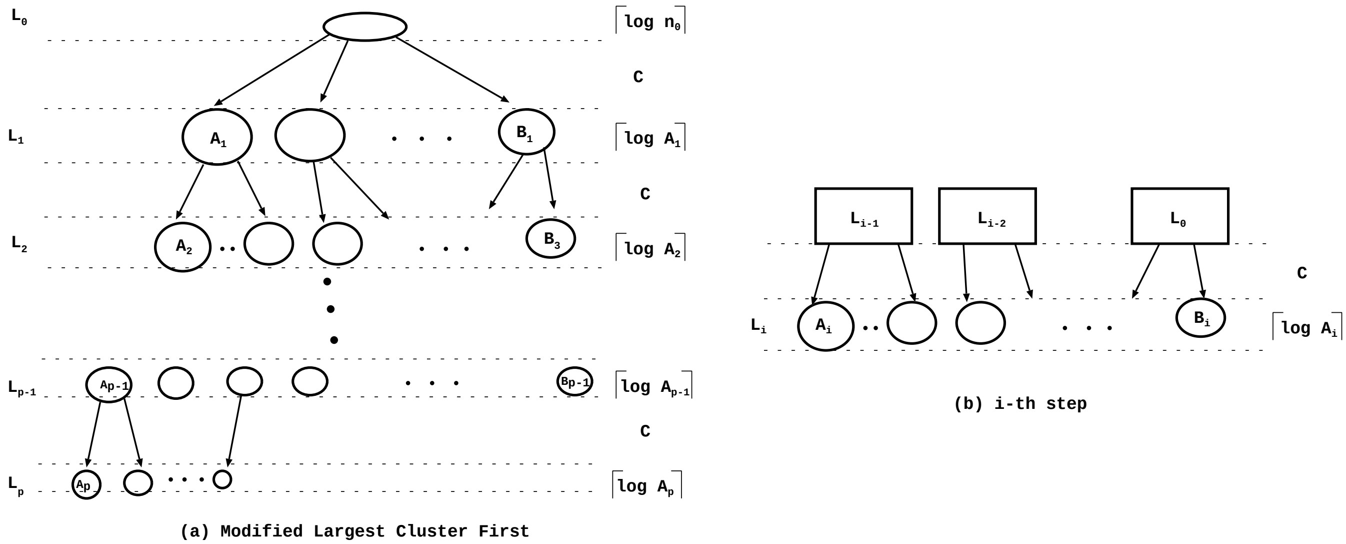

Figure 1: (a) It shows how *Modified LCF* works (b) At i-th step, all processors in clusters $K \in L_j$ $(j = 0, \ldots, i-1)$ send messages to L_i and then clusters in L_i perform local broadcasting. Therefore, i-th step takes $C + \lceil \log A_i \rceil$.

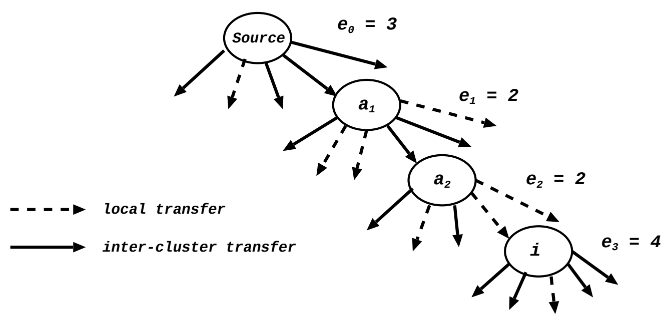

Figure 4: An example to show the inter-cluster transfers processor i experiences.

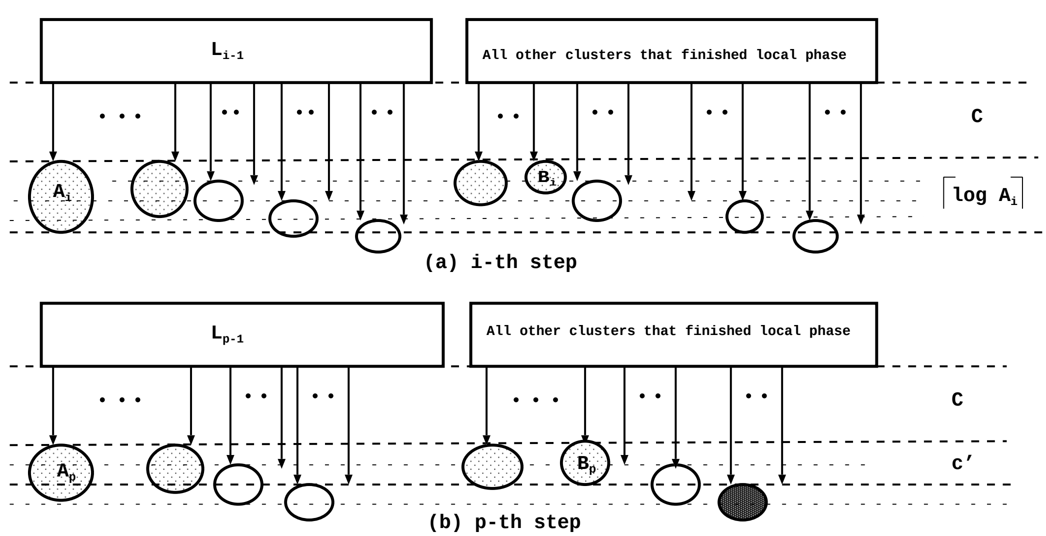

Figure 5: (a) It shows i-th step of LCF in postal model. In this example $C = 4$ so a processor can initiate (at most) 4 global transfers. Dotted clusters belong to L_i. (b) In p-th step, there can be some processors that receive messages later than A_p. The dark circle is the last cluster that receives the message.