Optimal Algorithms for Half-Duplex Inter-Group All-to-All Broadcast on Fully Connected and Ring Topologies

Qiao Kang, Ankit Agrawal, Alok Choudhary, Wei-keng Liao
EECS Department, Northwestern University
{qiao.kang, ankitag, choudhar, wkliao}@eecs.northwestern.edu

***Abstract*—Half-duplex inter-group collective communications are bipartite message transfer patterns such that the processes in a sender group pass messages to the processes in a receiver group. These communication patterns serve as basic operations for scientific application workflows. In this paper, we present optimal parallel algorithms for half-duplex inter-group all-to-all broadcast under bidirectional communication constraint on fully connected and ring topologies. We implement the algorithms using MPI communication functions and perform experiments on Cori. For the fully connected topology case, we compare our algorithms with production MPI libraries. For the ring topology case, we implement our proposed algorithms using MPI_Sendrecv function to emulate a ring topology environment. The proposed algorithms are compared with the intra-group Allgather algorithm emulated under the same environment. Message sizes ranging from 32KB to 4MB are used for evaluations. The proposed algorithms for fully connected topology are up to 5 times faster than the root gathering algorithm adopted by MPICH. The proposed algorithms for the ring topology are up to 1.4 times faster than the intra-group Allgather algorithm.**

***Index Terms*—MPI, Inter-group communication, Allgather**

I. Introduction

Half-duplex inter-group collective communications in Message Passing Interface (MPI)[1] are bipartite message transfer patterns such that the processes in a sender group transfer messages to the processes in a receiver group. Compared with intra-group collective communication, where all processes are both senders and receivers, half-duplex inter-group collective communication does not enforce the senders to receive any messages and the receivers to send any messages. Implementation of inter-group collective communication in theory should have smaller communication cost compared with intra-group collective communication given the same total number of processes, since the number of compulsory senders and receivers is less.

Argued in [2], inter-group collective communication is a basic communication pattern in scientific application workflows. For example, Liao et al.[3] propose a framework that allows parallel data transfer among workflow components in order to improve the performance of weather prediction systems SCALE-LETKF [4]. Hardware/Hybrid Accelerated Cosmology Code (HACC) [5], also has a demand for processing and transferring peta-byte sized data in real-time [6]. An advantage of using intergroup communication is fault tolerance. Examples discussed in [7] include applications such as DNA sequencing, graphics rendering and searching for extraterrestrial intelligence with a manager/worker model can benefit from the use of inter-communicators. To improve the performance of such systems, existing literatures have focused on reducing the data size of communications. For instance, Zhang et al. [8] propose a distributed framework that maximizes on-chip data exchange, which in turn reduces the communication frequency among components. Docan et al. [9] adopt an ActiveSpacing approach that moves programs to staging areas in order to reduce data exchange. Although reducing communication size and frequency can improve performance of workflow systems, scientific application performance can also benefit from optimal inter-group collective communication algorithms.

The most widely-used algorithm for inter-group communication is the root gathering algorithm, summarized in [10] and [11]. The root gathering algorithm has two stages: Single-process accumulation followed by one-to-all broadcast/scatter. MPICH [12] and OpenMPI [13], the most widely used MPI implementations in parallel processing community, adopt this strategy. However, the root gathering algorithm is not optimal because the receiving channels of the receivers are idle in at some stages.

In [2], we have proposed a full-duplex inter-group all-to-all broadcast algorithm on fully connected topology. In this paper, we present optimal algorithms for half-duplex inter-group all-to-all broadcast (Allgather) under bidirectional communication constraint for any number of senders and receivers on both fully connected and ring topologies. Unlike the root gathering algorithm, the proposed algorithms reduce the idle time of the receiving channels for the receivers. Moreover, the proposed algorithm for fully connected topology has more concise formulation and smaller startup latency compared with the full-duplex algorithm proposed in [2]. We provide detailed descriptions of algorithmic correctness and optimality.

For performance evaluation, we implement the proposed

algorithms for inter-group all-to-all broadcast using MPI communication functions. Experiments are conducted on Cori, a Cray XC40 supercomputer at the National Energy Research Scientific Computing Center (NERSC). Inter-group all-to-all broadcast can be achieved by the function MPI_Allgather using inter-group communicator. For fully connected topology case, direct comparisons with the function MPI_Allgather that adopts the root gathering algorithm are made. The communication network topology on Cori is pseudo fully connected (dragon-fly). We prove the point by evaluating and comparing our algorithms against the MPI library installed on Cori. For ring topology case, we implement our proposed algorithms using the MPI_Sendrecv function to emulate a ring topology environment. The proposed algorithms are compared with the intra-group Allgather algorithm emulated under the same environment. Message size used for evaluations ranges from 32KB to 4MB. The proposed algorithm for the fully connected topology is up to 5 times faster than the root gathering algorithm. The proposed algorithm for the ring topology is up to 1.4 times faster than the intra-group Allgather algorithm.

II. Background and Related work

The communication model used in this paper is based on the assumptions presented in [14], which are summarized below. Studies [15] have shown that these assumptions are widely assumed by collective communication algorithm designs.

1) **Parallel architecture**: An undirected and connected graph represents a network, where processes are vertices and links are edges. Processes can only send/receive messages to/from other processes if there are direct links between them.
2) **Bidirectional communication constraint**: When send or receive function at a process is called, the function is locked until the function returns. A process can receive and send messages at the same time.
3) **Communication cost**: Let t_w be communication transfer time per word and t_s be communication startup time. Sending a message of size k words from a sender to a receiver has communication cost $t_s + kt_w$. The term t_s is called startup latency and the term kt_w is called bandwidth.

A. Collective Communication

Collective communications defined by the MPI standard[1] have two categories: intra-group communication and inter-group communication. MPI communicator has an attribute that distinguishes these two categories.

Intra-group communication means that all processes are both senders and receivers. All processes in intra-group collective communication are symmetric. A process receives messages aggregated from the rest of processes in the end, though intermediate messages received by individual process may differ depending on the algorithm and topology.

Optimal intra-group communication algorithms have been well-studied [15]. Bertsekas [16] has proposes an optimal algorithm for all-to-all broadcast and total exchange on a hypercube topology. Thakur et al. [17] optimize intra-group Allgather using recursive doubling and Bruck algorithm [18] for non-power-of-2 number of processes.

Though intra-group communication are sufficient for supporting classical parallel computations such as matrix operations and prefix sums, demands for inter-group communication exists. Inter-group communication address the problem of bipartite message transfer. Instead of having a symmetric group of processes, processes are separated into two disjoint groups: one is the sender group and the other is the receiver group. The goal of inter-group communication is to deliver messages from the senders to the receivers. Although intra-group communication functions can achieve this goal by treating all processes as senders and receivers with the help of dummy messages, inter-group communication algorithms can have less communication cost compared with intra-group communication.

Existing works such as [10] and [11] present inter-group all-to-all broadcast (Allgather) implementations by extending MPI intra-group communication. All these works are based on the root gathering strategy: Single-process accumulation followed by one-to-all broadcast. MPICH [12] and OpenMPI [13], the most widely used MPI implementations in parallel processing community, use this strategy. However, the root gathering algorithm is not optimal because the receiving channels of the receivers are idle in some stages. Thus, algorithms that allow inter-group all-to-all broadcast without using a root process can be very useful.

Optimal communication cost for inter-group communication depends on the topology of processes. We present algorithms for two important topologies: Full connected and ring. Fully connected topology is a reasonable assumption for modern supercomputers. For example, the supercomputer Cori at NERSC is pseudo fully connected (dragon-fly), which means that the shortest distance between any two remote computing nodes is a constant value. Moreover, algorithms that uses binary indexing frequently assumes fully connected topology. For example, the multiple message broadcasting algorithm[19] assumes fully connected topology. Ring topology connects process with a small number of edges. It has the advantage of handling non-power of two number of processes. For example, intra-group Allgather and intra-group one-to-all broadcast functions implemented by MPICH adopt ring-based algorithm for handling non-power of two number of processes. Moreover, some systems are configured as multi-dimensional rings (N-dimensional torus). The optimal algorithm for ring topology can be readily extended to multi-dimensional ring.

III. Algorithms for Fully Connected Topology

We first define the mathematical notations in this paper. Let $A = \{a_0, ..., a_{p-1}\}$ and $B = \{b_0, .., b_{q-1}\}$ be two disjoint arrays of processes. A denotes the set of sender ranks of size p. B denotes the set of receiver ranks of size q. Processes in A are labeled with ranks from 0 to $p-1$. Processes in B are labeled with ranks from p to $p+q-1$. Initially, every $a_i \in A$ has a unique message m_i of size k words. The goal is to let b_j $\forall 0 \leq j < q$ receive m_i $\forall 0 \leq i < p$.

The theoretical lower bound for bandwidth can be established using the minimum time required for a single receiver to receive all messages. A receiver can receive p number of messages in a single step or multiple steps. Nevertheless, the total transfer time for bipartite communication takes at least pkt_w. The time taken for all processes to receive required messages is longer than the time taken for a specific process to receive required message, so pkt_w is a theoretical lower bound for bandwidth.

The theoretical lower bound for startup latency can be established using one-to-all broadcast from one sender to all receivers and all-to-one gather from all senders to one receiver, which are both sub-problems of inter-group all-to-all broadcast. Algorithms for one-to-all broadcast require at least $\log(q+1)$ steps. Since $\log(q+1) \leq \log(2q) = 1+\log(q) < 1+\log(q+1)$, $1+\log(q)$ is a tight lower bound for startup latency. Alternatively, a receiver receives messages aggregated from p senders. Aggregating messages from p processes, requires at least $\log(p+1)$ steps, so $\log(p)+1$ is another tight lower bound for optimal startup latency. Achieving either of the lower bound for startup latency would justify the optimality of startup latency of the proposed algorithms. Since $p = q$ and the proposed algorithm has $\log(p)+1$ number of steps, the startup time is optimal.

Suppose $p = q$, the proposed algorithm can be divided into two phases. Message m_j is transfered from a_j to b_j $\forall j \in [0, p-1]$. This phase has communication cost $(t_s + kt_w)$. Then, the receivers perform intra-group Allgather over messages m_j $\forall j \in [0, p-1]$. This phase has communication cost $\log(p)\, t_s + (p-1)\, kt_w$. The total communication cost is $(1+\log(p))\, t_s + pkt_w$. It is obvious that all receivers receive messages from all senders.

In real-world applications, the number of senders and receivers are not necessarily equal, so strategies for handling $p \neq q$ are proposed.

When $p > q$, q number of processes in group A concurrently send messages to group B. The communication cost is $t_s + kt_w$. The remaining $p-q$ number of processes in the sender group joins the q processes in the receiver group, forming a group of size p. The p processes spanned across two groups perform intra-group Allgather. The communication cost of this process is $\log(p)\, t_s + (p-1)\, kt_w$. The total communication cost is $(1+\log(p))\, t_s + pkt_w$, which is the same as $p = q$ case. As a result, the algorithm is optimal. Figure 1a and 1b illustrate an example when $p = 6$ and $q = 2$. Although processes in group A are aggregating unnecessary messages to themselves during intra-group Allgather, the aggregated messages save the number of steps for the receivers to receive all messages. To elaborate on the necessity of using the sender channels, we consider an alternative algorithm that passes all messages from the senders to the receivers with $\frac{p}{q}$ number of steps. Then the receivers perform intra-group Allgather by themselves. Although this algorithm does not send redundant messages, it has at least $\frac{p}{q}$ of startup latency, which is not optimal for small message sizes.

When $p < q$, the algorithm consists of two stages. In the first stage, the processes $\{a_i, b_{i+pj} : 0 \leq j < \frac{q}{p}\}$ $\forall i \in [0, p-1]$ form broadcast groups with root a_i. In every broadcast group, the root broadcasts its message to all other processes. In the second stage, the processes $\{b_{ip+j} : 0 \leq j < p\}$ $\forall i \in \left[0, \frac{q}{p} - 1\right]$ form subgroups of the receiver group, each with p number of processes. All subgroups of the receiver group execute intra-group Allgather independently. The one-to-all broadcasting has communication cost $2\log\left(\frac{q}{p}+1\right) t_s + 2kt_w$ using the multiple message broadcasting algorithm[19]. This communication cost for one-to-all broadcast is also claimed by the latest version of MPICH. The subgroup all-to-all broadcast has communication cost $\log(p)\, t_s + (p-1)\, kt_w$. Total communication cost is $\left(\log(p)) + 2\log\left(\frac{q}{p}+1\right)\right) t_s + (p+1)\, kt_w$. When p and q are large enough, the total communication cost converge to $\log(q)\, t_s + pkt_w$. Hence the total communication cost is optimal. Figure 1c and 1d illustrate an example when $p = 2$ and $q = 6$.

A. Comparison with Root Gathering Algorithm

The root gathering algorithm for inter-group all-to-all broadcast, adopted by both MPICH and OpenMPI, consists of three stages. In the first stage, processes in the sender group use intra-group Gather function to accumulate all messages to a root sender. Without loss of generality, we use a_0 to denote this root sender. This stage has communication cost $\log(p)\, t_s + (p-1)\, kt_w$. In the second stage, a_0 send p messages aggregated in the previous stage to the root $b_0 \in B$. This stage has communication cost $t_s + pkt_w$ Finally, root b_0 uses one-to-all broadcast function to pass accumulated messages to all receivers. Since the message size of broadcast is pk, this stage has communication cost $2\log(q+1)\, t_s + 2pkt_w$. Therefore, the total communication cost is $(\log(p) + 2\log(q))\, t_s + 4pkt_w$ for large p and q.

Table I summarizes the communication cost of the proposed algorithm and the root gathering algorithm. When $p \geq q$, the startup latency of the proposed algorithm is $2\log(q)$ steps less than the root gathering algorithm. When $p < q$, the startup latency of the proposed algorithm is $\log(q) + \log(p)$ steps less than the root gathering algorithm. The bandwidth of the proposed methods is approxi-

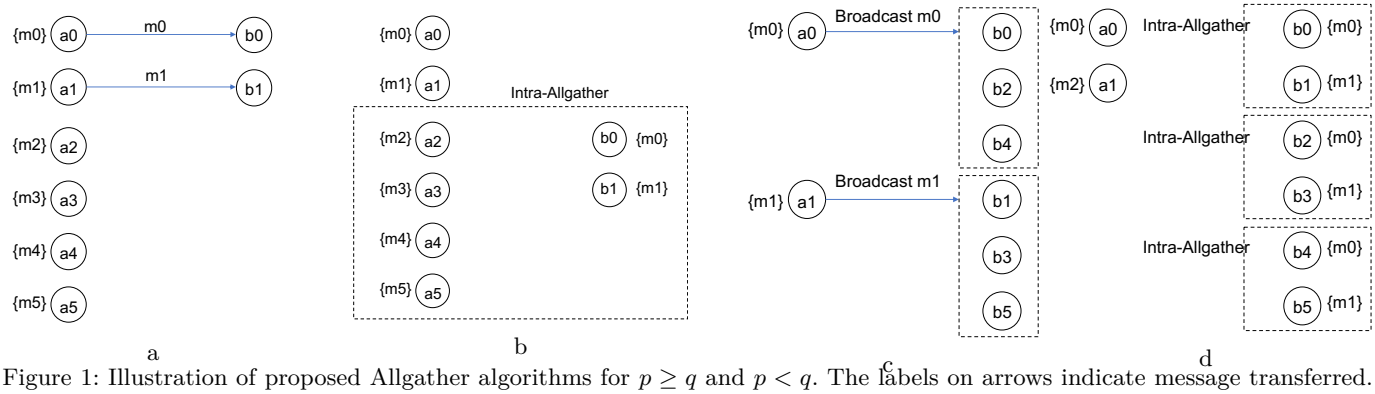

Figure 1: Illustration of proposed Allgather algorithms for $p \geq q$ and $p < q$. The labels on arrows indicate message transferred. (a) First step for $p \geq q$. (b) Second step for $p \geq q$. (c) First step for $p < q$. (d) Second step for $p < q$.

Table I: Comparison of theoretical communication cost for the proposed algorithm and the root gathering algorithm when the number of senders is p and the number of receivers is q on fully connected network.

Method	Startup Latency/t_s	Bandwidth/kt_w
Root Gathering	$\log(p) + 2\log(q)$	$4p$
Proposed $p \geq q$	$\log(p)$	p
Proposed $p < q$	$\log(p)$	p

mately 4 times faster than the root gathering algorithm for any p, q. Since $q+1$ and $\frac{p}{q}+1$ can be non-power of two, the exact difference depends on the implementation strategies. Nevertheless, the proposed algorithm is a constant time faster than the root gathering algorithm.

IV. Algorithms for Ring Topology

Ring topology connects processes with a small number of links. Rank i has two edges: One is connected to rank $(i-1) \bmod (p+q)$ and the other is connected to rank $(i+1) \bmod (p+q)$. We use ranks $0, .., p-1$ to denote the senders and ranks $p, .., p+q-1$ to denote the receivers. One important assumption is that p and q are both even number for convenience of formulation. If they are not, a dummy process can be used to make up to the next even number.

The proposed optimal inter-group all-to-all broadcast algorithms on ring topology contain three stages. Every stage consists of multiple steps. The steps are send/receive functions executed in parallel. Algorithm 1 describes all stages and steps of the algorithm when $p = q$. The i loops represent the stages. The j loops are the parallel steps. The concurrent labels in the algorithm refer to the ID of message sequence defined as the following.

We briefly discuss the high-level ideas of Algorithm 1 for transferring messages from the senders to the receivers. Like the intra-group Allgather algorithm for ring topology, where messages are circulated in a "pipeline" movement for $p+q-1$ steps, the proposed algorithm creates multiple "pipelines" of messages in different directions. However, naive pipelining of messages from both sides (via process 0 and $p-1$) of the ring will break bidirectional communication constraint mentioned in section 2, due to the limited number of connections. The problem is more complicated for $p \neq q$ cases. The design of the proposed algorithms solves this issue by introducing the concept of message sequence in Definition IV.1, a formal description of pipeline movements of messages. These message sequences are controlled by the proposed algorithms so that their movements do not violate the bidirectional communication constraint using interleaved indexing. Nevertheless, they can fulfill the objectives of half-duplex all-to-all broadcast with optimal communication cost.

Definition IV.1. *A message sequence over an array of messages $m_i, .., m_{i+k}$ in a parallel communication algorithm is defined to be parallel steps such that the list of receivers for every element of the message array has either increasing or decreasing index order for all steps.*

In the rest of this section, we present complete details of theoretical analysis for all cases. We suggest readers seeking high-level insights to go through Figures 2 and 3 first. Algorithms 1 and 3 provide details about process indexing that can be readily used for implementations. Section 4.1-4.4 and the second half of 4.5 should be the interest of people who favor comprehensive proofs of algorithms algebraically.

We prove that Algorithm 1 is feasible with respect to three constraints. The first constraint, referred as message availability, is that any message sent is available at the sender. The second constraint is the bidirectional communication constraint mentioned in section 2. The third constraint is message completeness, which means that every receiver receives all sender messages in the end.

A. Message availability

Theorem IV.2. *Algorithm 2 satisfies message availability constraint if process $p_{(a+cj+ki) \bmod (p+q)}$ has message $m_{(j+b) \bmod (p+q)}$ at the beginning $\forall j \in \{0, ..., x\}$. (Message continuity).*

Proof. The sender always becomes the receiver in the next iteration of i loop for any j. Hence message availability constraint is satisfied. □

In algorithm 1, there are three message sequences (labeled with #). It is possible to show that they all obey message availability constraint.

Algorithm 1: Optimal inter-group all-to-all broadcast on Ring Topology ($p = q$).

for $i \in \{0, ..., \frac{p}{2} - 1\}$ **do**
  #Concurrent #1
  for $j \in \{\frac{p}{2}, ..., p - 1\}$ **do**
    $j + i$ send m_j to $j + i + 1$
  end
  #Concurrent #2
  for $j \in \{0, ..., \frac{p}{2} - 1\}$ **do**
    $(j - i) \bmod (p + q)$ send m_j to $(j - i - 1) \bmod (p + q)$
  end
  #Concurrent #3
  for $j \in \{0, ..., i - 1\}$ **do**
    $\frac{p}{2} - i + 2j$ send $m_{\frac{p}{2}-i+j}$ to $\frac{p}{2} - i + 2j + 1$
  end
end
for $i \in \{0, ..., \frac{p}{2} - 1\}$ **do**
  #Concurrent #1
  for $j \in \{\frac{p}{2}, ..., p - 1\}$ **do**
    $\frac{p}{2} + j + i$ send m_j to $\frac{p}{2} + j + i + 1$
  end
  #Concurrent #3
  for $j \in \{0, ..., \frac{p}{2} - 1\}$ **do**
    $2j + i$ send m_j to $2j + i + 1$
  end
end
for $i \in \{0, ..., \frac{p}{2} - 1\}$ **do**
  #Concurrent #1
  for $j \in \{\frac{p}{2}, ..., p - 1\}$ **do**
    if $p + j + i + 1 < 2p$ **then**
      $p + j + i$ send m_j to $p + j + i + 1$
    end
  end
  #Concurrent #2
  for $j \in \{1, ..., i\}$ **do**
    $\frac{3p}{2} - i + 2j - 1$ send m_{j-1} to $\frac{3p}{2} - i + 2j - 2$
  end
  #Concurrent #3
  for $j \in \{0, ..., \frac{p}{2} - 1\}$ **do**
    $\frac{p}{2} + i + 2j$ send m_j to $\frac{p}{2} + i + 2j + 1$
  end
end

Algorithm 2: Message Continuity

Data: $k \in \{1, -1\}$,$x, a, b, c \in \mathbb{Z}^+$
for $i \in \{0, ..., \frac{p}{2} - 1\}$ **do**
  for $j \in \{0, ..., x\}$ **do**
    $p_{(a+cj+ki) \bmod (p+q)}$ send $m_{(j+b) \bmod (p+q)}$ to $p_{(a+cj+k(i+1)) \bmod (p+q)}$
  end
end

For sequence #1 and #2, Theorem IV.2 justifies the message availability constraint by concatenating the i loops.

For sequence #3, we can examine all i loops. For the first i loop, let m_c be any arbitrary constant rank of message in message array of sequence #3. We have a linear system $c = \frac{p}{2} - i + j$ with variables i and j. Let i_0 be the very first iteration of i loop when m_c is transfered in message sequence # 3. $i_0 = \frac{p}{2} - c$ is minimum when $j = 0$. Processor $\frac{p}{2} - i_0$ has $m_{\frac{p}{2}-i_0}$ by definition. Thus, the base case for m_c is established. When m_c is sent at the $i_k \geq i_0$ iteration,he receiver of m_c is rank $\frac{p}{2} - i_k + 2j_k + 1 = c + j_k + 1$ for $j = j_k$ in this iteration. In the $i_k + 1$ iteration, $j = j_k + 1$ is solved from the definition of c. The sender of m_c is exactly process $\frac{p}{2} - (i_k + 1) + 2(j_k + 1) = c + j_k + 1$. Hence the induction of message availability of m_c is completed. Consider the receiver ranks in the last iteration of the first stage, $\frac{p}{2} - \left(\frac{p}{2} - 1\right) + 1 + 2j = 2j + 2$ and j takes values from 0 to $\frac{p}{2} - 2$, so the senders at the beginning of the second stage are the receivers in the last iteration of first stage loop except process 0. By definition, process 0 has message m_0. Hence all messages sent at the start of the second stage are available. Message continuity theorem can be used to show that the second i loop fulfills the message availability constraint. For the third i loop, it continues the second loop for the range of i from $\frac{p}{2}$ to $p - 1$. According to Theorem IV.2, sequence #3 is message complete.

B. Bidirectional communication constraint

Single-port communication constraint is not violated if the following two conditions are true. Every sender is unique within an i loop. Moreover, every receiver is unique within an i loop. The uniqueness across multiple j loops within an i loop can be verified by computing the ranges of sender/receiver ranks. If the ranges of sequence ranks do not overlap, the senders and receivers must be different for any i. The ranges of sequence #1 and sequence #2 in the first i loop do not overlap. The ranges of sequence #3 and sequence #2 in the first i loop do not overlap.

When the ranges of senders and receivers in two sequences under an i loop overlap, proof by contradiction can be used to show uniqueness of ranks given the same arbitrary value of i. Suppose a sender sends a message to different processes within an i loop. There exist two integers j_1 and j_2 within the ranges of j loops such that the ranks of senders are equal given the same i. The same reasoning can be applied to receivers. We can enumerate the rest pairs of sequences.

Consider sequence #1 and sequence #3 in the first i loop. For senders, we have $j_1 + i = \frac{p}{2} - i + 2j_2$. It follows that $j_1 = \frac{p}{2} - 2i + 2j_2$. However, $j_2 < i$ by definition. Hence $j_1 < \frac{p}{2}$, which is outside its range $\left[\frac{p}{2}, p - 1\right]$. The same argument works for receivers because receiver ranks are sender ranks plus 1.

Consider sequence #1 and sequence #3 in the second i loop. For senders, we have $\frac{p}{2} + j_1 + i = 2j_2 + i$. It follows that $j_1 = 2j_2 - \frac{p}{2}$. j_2 takes values from 0 to $\frac{p}{2} - 1$. Hence $j_1 < \frac{p}{2}$, which is outside the range $\left[\frac{p}{2}, p - 1\right]$. The same argument works for receivers because receiver ranks are sender ranks plus 1.

Consider sequence #1 and sequence #2 in the third i loop. For senders, we have $p + j_1 + i = \frac{3p}{2} - i + 2j_2 - 1$. It follows that $j_1 = \frac{p}{2} + 2j_2 - 2i - 1$. $j_2 \leq i$ implies that $j_1 \leq \frac{p}{2} - 1$. However, $j_1 \geq \frac{p}{2}$ by definition. Therefore, no solution exists for j_1. For receivers, we have $p + j_1 + i + 1 = \frac{3p}{2} - i + 2j_2 - 2$, it follows that $j_1 = \frac{p}{2} + 2j_2 - 2i - 3$. $j_2 \leq i$ implies that $j_1 \leq \frac{p}{2} - 3$. However, $j_1 \geq \frac{p}{2}$ by definition. Therefore, solution for j_1 does not exist.

Consider sequence #1 and sequence #3 in the third i loop. For senders, we have $p + j_1 + i = \frac{p}{2} + i + 2j_2$. It follows that $j_1 = 2j_2 - \frac{p}{2}$. j_2 takes values from range 0 to $\frac{p}{2} - 1$, so $j_1 \leq \frac{p}{2} - 1$. However, j_1 takes values in $\left[\frac{p}{2}, p - 1\right]$. The same argument works for receivers because receiver ranks are sender ranks plus 1.

Consider sequence #2 and sequence #3 in the third i loop. For senders, we have $\frac{3p}{2}-i+2j_1-1=\frac{p}{2}+i+2j_2$. It follows that $2\left(j_2-j_1+i\right)=p-1$. By assumption, p is an even number and i is an integer, so integer solutions for j_1 and j_2 do not exist. For receivers, we have $\frac{3p}{2}-i+2j_1-2=\frac{p}{2}+i+2j_2+1$. It follows that $2\left(j_2-j_1+i\right)=p-3$. Integer solutions for j_1 and j_2 do not exist.

C. Completeness

Completeness means that all messages from senders are received by all receivers when the algorithm finishes.

Theorem IV.3. *Let M_1 be message sequence over message array $m_i,..,m_{i+c}$ and M_2 be message sequence over message array $m_{i+c},..,m_i$ such that the receivers of M_1 and M_2 have reverse index order. If $\forall j \in [i,i+c]$, m_j in both sequences is received by some pair of adjacent receivers (m_j intersects in both sequences), process set P' that contains processes between the front senders of M_1 and M_2 is message complete.*

Proof. Suppose that $\forall j \in [i,i+c]$, m_j is received by adjacent receivers $v-1$ and v in both sequences. Without loss of generality, suppose M_1 has receivers in increasing index order at all steps and M_2 has receivers in decreasing index order at all steps. By definition of message sequence, when $v-1$ receives m_j in M_1, all processes in P' with ranks less than $v-1$ have received message m_j. When v receives m_j in M_2, all processes in P' with ranks greater than v have received message m_j. Therefore, P' is message complete over the message array. □

For sequence #1, it is clear that it travels through all the receivers by combining all the three stages. Hence receivers are message complete for messages $\{m_j : j \in \left[\frac{p}{2}, p-1\right]\}$.

For messages $\{m_j : j \in \left[0, \frac{p}{2}-1\right]\}$, we can show that the message m_j in sequence #2 and sequence #3 intersects at some receiver. Because all receivers are between the front senders of the two sequences, the theorem justifies message completeness. There is no stop condition in both sequences, so all messages in the sequence are transferred without delay. Therefore, m_j moves exactly $\frac{p}{2}+\frac{p}{2}-1-j=p-1-j$ hops in sequence #2 to decreasing index direction. The final receiver of m_j in sequence #2 is $(j-(p-1-j)) \bmod 2p = 2j+p+1$. For message sequence #3, message m_j at the start of stage 2 travels p hops in increasing direction. The final receiver of m_j in sequence #3 is $2j+p$. Hence the final receiver differs by 1 index, which means that the sequences intersect. It follows from Theorem IV.3 that all receivers have received messages $\{m_j : j \in \left[0, \frac{p}{2}-1\right]\}$.

D. Optimality

The optimal algorithm for inter-group all-to-all broadcast on ring topology that contains p senders with lower ranks and q receivers with higher ranks has at least $k\left(p+\frac{q}{2}-1\right)t_w$ bandwidth. This statement can be proved by considering the receiver at rank $p+\frac{q}{2}-1$ and $p+\frac{q}{2}$. Both of them need to receive p messages. Moreover, it takes at least $\frac{q}{2}-1$ steps for the first message to arrive at either of them. As a result, the communication cost is at least $\left(p+\frac{q}{2}-1\right)(kt_w+t_s)$.

The total communication cost of Algorithm 1 is the number of i iterations multiplied by t_s+t_w. Hence the communication cost is $\left(p+\frac{q}{2}\right)\left(t_s+t_w\right)k$, which converges to optimal bandwidth bound $\left(p+\frac{q}{2}-1\right)kt_w$ for large p,q.

Table II: Comparison of communication cost for the proposed algorithm and intra-group Allgather algorithm when the number of senders is p and the number of receivers is q on ring network.

Method	Startup latency/t_s	Bandwidth/kt_w
Intra-Allgather	$p+q-1$	$p+q-1$
Proposed	$p+\frac{q}{2}$	$p+\frac{q}{2}$

Table II summarizes the complexities of communication cost compared with intra-group Allgather algorithm. The improvement of communication cost is $\frac{p+q-1}{p+\frac{q}{2}}$. Based on this ratio, we expect that the improvement increases as the number of receivers increases.

E. Imbalanced Graph

In this section, we discuss methods for handling imbalanced numbers of senders and receivers. The three main stages of the Algorithm 1 remain unchanged.

When $p>q$, the principle is the same as $p=q$ case. The directions for all three sequences remain the same. The length of sequence #1 is $p-\frac{q}{2}$ over message array $\{m_{\frac{q}{2}},..,m_{p-1}\}$. The length of sequence #2 and #3 is $\frac{q}{2}$ over message array $\{m_0,..,m_{\frac{q}{2}-1}\}$. The first and third stage have $\frac{q}{2}$ steps. The second stage has $p-\frac{q}{2}$ steps. The previous proofs based on the three sequences are valid. The total communication cost is $\left(p+\frac{q}{2}\right)\left(t_s+t_w\right)k$. Hence the algorithm is optimal. Figure 3 illustrates an example when $p=4$ and $q=6$.

When $p<q$, as long as the process $p+q-1$ receives any messages delivered by sequence #1, there is at least a $q\left(t_s+t_w\right)k$ component in the total communication cost. If q is significantly larger than p, the communication cost is not optimal. Optimal communication cost can be reached by adding a sequence #4 that is symmetric to sequence #3 over messages $\{m_{\frac{p}{2}-1},..,m_0\}$ in the opposite direction at every stage. Algorithm 3 is a modified version for handling $p<q$ case. For message availability, the proof for sequence #1, #2, and #3 in previous section can be reused. Sequence #4 is symmetric to sequence #3 in opposite direction, so message availability is ensured. Figure 2 illustrates an example when $p=6$ and $q=4$.

For bidirectional constraint, we show that the new sequence does not conflict with existing sequences. Sequence #4 is divided into two parts in the second i loops. The computing of $\frac{i}{2}$ takes ceiling of the result. The range of sequence #4-1 does not overlap the ranges of sequence #1 and #2 in the second stage. Moreover, the range of sequence #3 does not overlap the ranges sequence #2 and sequence #4-2 in the second stage.

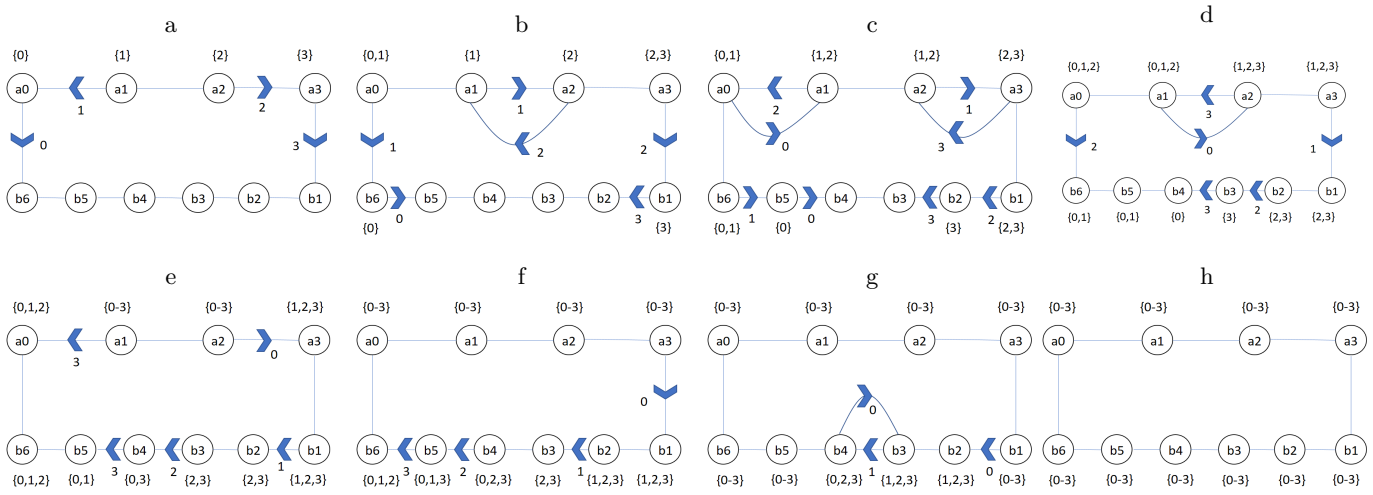

Figure 2: An example (left to right and top to bottom) of the proposed algorithm for the ring topology all-to-all broadcast when $p = 4$ and $q = 6$. Message available at every step is labeled above/below a process. Arrows illustrate messages transferred. The total number of steps is 7.

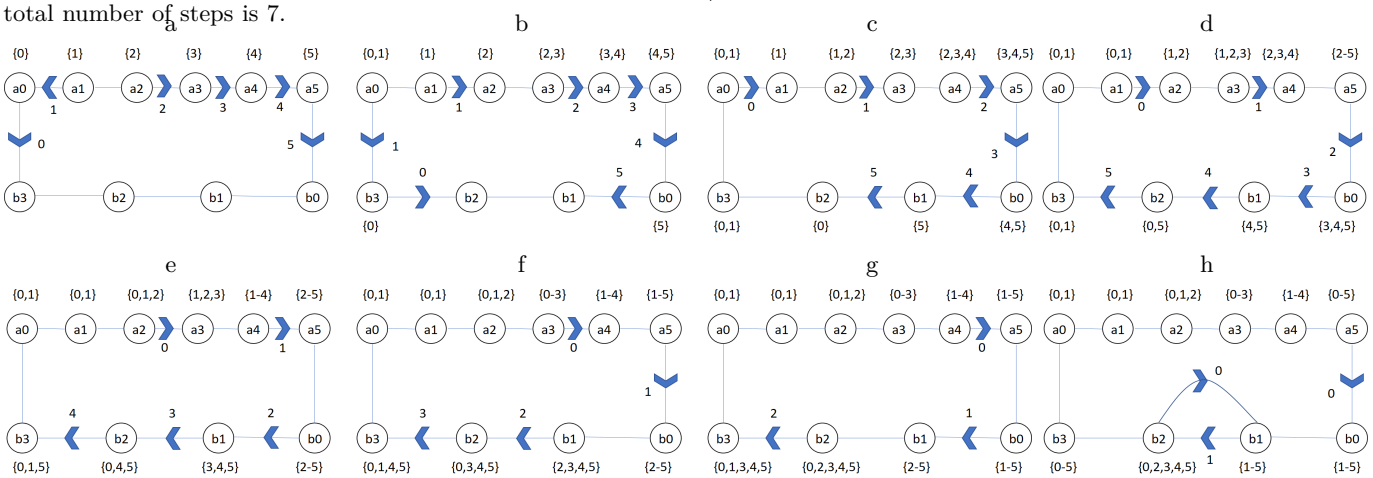

Figure 3: An example (left to right and top to bottom) of the proposed algorithm for the ring topology all-to-all broadcast when $p = 6$ and $q = 4$. Message available at every step is labeled above/below a process. Arrows illustrate messages transferred. The total number of steps is 8.

Consider sequence #1 and sequence #4 in the first i loop. For senders, we have $j_1 + i = 2j_2 - i - \frac{p}{2} + 1$. It follows that $j_1 = 2j_2 - 2i - \frac{p}{2} + 1$. Since $j_2 \leq \frac{p}{2} + i - 1$, $j_1 \leq \frac{p}{2} - 1$, which is out of the range of j in sequence #1. For receivers, we have $j_1 + i + 1 = 2j_2 - i - \frac{p}{2}$, it follows that $j_1 = 2j_2 - 2i - \frac{p}{2} - 1$. Since $j_2 \leq \frac{p}{2} + i - 1$, $j_1 \leq \frac{p}{2} - 3$, which is out of the range of j in sequence #1. The same argument works for receivers because receiver ranks are sender ranks plus 1.

Consider sequence #2 and sequence #4 in the first i loop. For senders, we have $j_1 - i = 2j_2 - i - \frac{p}{2} + 1$. It follows that $j_1 = 2j_2 - \frac{p}{2} + 1$. Since $j_2 \geq \frac{p}{2}$, $j_1 \geq \frac{p}{2} + 1$, which is out of the range of j in sequence #2. The same argument works for receivers because receiver ranks are sender ranks minus 1.

Consider sequence #3 and sequence #4 in the first i loop. For senders, we have $\frac{p}{2} - i + 2j_1 = 2j_2 - i - \frac{p}{2} + 1$. It follows that $2j_1 = 2j_2 - p + 1$. By assumption, p is even. Hence $2j_2 - p + 1$ is odd. Integer solutions for j_1 do not exist. For receivers, we have $\frac{p}{2} - i + 2j_1 + 1 = 2j_2 - i - \frac{p}{2}$. It follows that $2j_1 = 2j_2 - p - 1$. By assumption, p is even. Hence $2j_2 - p - 1$ is odd. Integer solutions for j_1 and j_2 do not exist.

Consider sequence #3 and sequence #4-1 in the second i loop. For senders, we have $2j_1 + i = 2j_2 - i - p + 1$. It follows that $2(j_2 - j_1) = 2i + p - 1$. By assumption, p is even. Hence $2i+p-1$ is odd. Integer solutions for j_1 and j_2 do not exist. For receivers, we have $2j_2 - i - p = 2j_1 + i + 1$. It follows that $2(j_2 - j_1) = 2i + p + 1$. By assumption, p is even. Hence $2i + p + 1$ is odd. Integer solutions for j_1 and j_2 do not exist.

Consider sequence #1 and sequence #4-2 in the second i loop. For senders, we have $\frac{p}{2} + j_1 + i = 2j_2 - i + q + 1$. It follows that $j_1 = 2j_2 - 2i + q - \frac{p}{2} + 1$. The condition enforces $2j_2 > 2i + q - \frac{p}{2} > 2i + \frac{p}{2}$, so $j_1 > p + 1$, which is out of the range of j in sequence #1. For receivers, we have $\frac{p}{2} + j_1 + i + 1 = 2j_2 - i + q$. It follows that $j_1 = 2j_2 - 2i + q - \frac{p}{2} - 1$. The condition branch enforces $2j_2 > 2i + q - \frac{p}{2} > 2i + \frac{p}{2}$,

Algorithm 3: Optimal inter-group all-to-all broadcast on Ring Topology (p<q)

for $i \in \{0, ..., \frac{p}{2} - 1\}$ **do**
  #Concurrent #1
  for $j \in \{\frac{p}{2}, ..., p-1\}$ **do**
    $j + i$ send m_j to $j + i + 1$
  end
  #Concurrent #2
  for $j \in \{0, ..., \frac{p}{2} - 1\}$ **do**
    $(j - i) \bmod (p + q)$ send m_j to $(j - i - 1) \bmod (p + q)$
  end
  #Concurrent #3
  for $j \in \{0, ..., i - 1\}$ **do**
    $\frac{p}{2} - i + 2j$ send $m_{\frac{p}{2}-i+j}$ to $\frac{p}{2} - i + 2j + 1$
  end
  #Concurrent #4
  for $j \in \{\frac{p}{2}, ..., \frac{p}{2} + i - 1\}$ **do**
    $2j - i - \frac{p}{2} + 1$ send m_j to $2j - i - \frac{p}{2}$
  end
end
for $i \in \{0, ..., \frac{q}{2} - 1\}$ **do**
  #Concurrent #1
  for $j \in \{\frac{p}{2}, ..., p-1\}$ **do**
    $\frac{p}{2} + j + i$ send m_j to $\frac{p}{2} + j + i + 1$
  end
  #Concurrent #2
  for $j \in \{0, ..., \frac{p}{2} - 1\}$ **do**
    if $i < \frac{q}{2} - \frac{p}{2}$ **then**
      $\frac{p}{2} + q - i + j$ send m_j to $\frac{p}{2} + q - i + j - 1$
    end
  end
  #Concurrent #3
  for $j \in \{0, ..., \frac{p}{2} - 1\}$ **do**
    $2j + i$ send m_j to $2j + i + 1$
  end
  #Concurrent #4-1
  for $j \in \{\frac{p}{2} + \frac{i}{2}, ..., p-1\}$ **do**
    $2j - i - p + 1$ send m_j to $2j - i - p$
  end
  #Concurrent #4-2
  for $j \in \{\frac{p}{2}, ..., \frac{p}{2} + \frac{i}{2} - 1\}$ **do**
    if $2j > 2i + \frac{p}{2}$ **then**
      $2j - i + q + 1$ send m_j to $2j - i + q$
    end
  end
end
for $i \in \{0, ..., \frac{p}{2} - 1\}$ **do**
  #Concurrent #1
  for $j \in \{\frac{p}{2}, ..., p-1\}$ **do**
    if $j + i + 1 < \frac{p}{2} + \frac{q}{2}$ **then**
      $\frac{p}{2} + \frac{q}{2} + j + i$ send m_j to $\frac{p}{2} + \frac{q}{2} + j + i + 1$
    end
  end
  #Concurrent #2
  for $j \in \{1, ..., i\}$ **do**
    $\frac{q}{2} + p - i + 2j - 1$ send m_{j-1} to $\frac{q}{2} + p - i + 2j - 2$
  end
  #Concurrent #3
  for $j \in \{0, ..., \frac{p}{2} - 1\}$ **do**
    $\frac{q}{2} + i + 2j$ send m_j to $\frac{q}{2} + i + 2j + 1$
  end
end

so $j_1 > p - 1$, which is out of the range of j in sequence #1.

Consider sequence #2 and sequence #4-2 in the second i loop. For senders, we have $\frac{p}{2} + q - i + j_1 = 2j_2 - i + q + 1$. It follows that $j_1 = 2j_2 - \frac{p}{2} + 1$. Since $j_2 \geq \frac{p}{2}$, $j_1 \geq \frac{p}{2} + 1$, which is outside the range of j in sequence #2. The same argument works for receivers because receiver ranks are sender ranks minus 1.

Consider sequence #2 and sequence #1 in the second i loop. For senders, we have $\frac{p}{2} + q - i + j_1 = \frac{p}{2} + i + j_2$. It follows that $j_2 = q - 2i + j_1$. Since $i < \frac{q}{2} - \frac{p}{2}$ by the conditional branch and i is an integer, $2i < q - p - 1$. It follows that $j_2 > p + j_1 + 1$. However, $j_1 \geq 0$, so $j_2 > p + 1$, which is out of range of j in sequence #1. For receivers, we have $\frac{p}{2} + q - i + j_1 - 1 = \frac{p}{2} + i + j_2 + 1$. It follows that $j_2 = q - 2i + j_1 - 2$. Since $i < \frac{q}{2} - \frac{p}{2}$ by the conditional branch and i is an integer, $2i < q - p - 1$. It follows that $j_2 > p + j_1 - 1$. However, $j_1 \geq 0$, so $j_2 > p - 1$, which is out of range of j in sequence #1.

Proof for message completeness can be divided into two parts: $\{m_j : j \in \left[0, \frac{p}{2} - 1\right]\}$ and $\{m_j : j \in \left[\frac{p}{2}, p\right]\}$.

For messages $\{m_j : j \in \left[0, \frac{p}{2} - 1\right]\}$, we can show that the message m_j in sequence #2 and sequence #3 intersects at some receiver. m_j moves exactly $\frac{p}{2} + \left(\frac{q}{2} - \frac{p}{2}\right) + \left(\frac{p}{2} - 1 - j\right) = \frac{p}{2} + \frac{q}{2} - 1 - j$ hops in sequence #2 to decreasing index direction. m_j moves exactly $j + \frac{q}{2} + \frac{p}{2}$ hops in sequence #3 to increasing index direction. The final receiver of m_j in sequence #2 is $\left(j - \left(\frac{p}{2} + \frac{q}{2} - 1 - j\right)\right) \bmod (p + q) = 2j + \frac{p}{2} + \frac{q}{2} + 1$. The final receiver of m_j in sequence #3 is $2j + \frac{q}{2} + \frac{p}{2}$. Hence the final receiver differs by 1 index, which means that all messages intersect. It follows from Theorem IV.3 that all receivers have received messages $\{m_j : j \in \left[0, \frac{p}{2} - 1\right]\}$.

For $\{m_j : j \in \left[\frac{p}{2}, p - 1\right]\}$, we can show the message m_j in sequence #1 and sequence #4 intersects at some receiver for any $j \in \left[0, \frac{p}{2} - 1\right]$ or m_j in sequence #1 reaches $p + q - 1$ at the end of the algorithm. If m_j is forced to stop at some receiver by the condition in sequence #4-2, it would intersect message m_{p-1} in the second i loop. Because there are $\frac{p}{2}$ steps that sequence #1 will move forward in stage 3, m_j in sequence #1 must intersect m_j in sequence #4. On the other hand, if the condition in sequence #4-2 is not triggered for m_j, m_j travels as far as $\frac{q}{2} + \frac{p}{2} - \left(j - \frac{p}{2} + 1\right)$ hops, so the final receiver is $\left(j - \left(\frac{q}{2} + \frac{p}{2} - \left(j - \frac{p}{2} + 1\right)\right)\right) \bmod (p + q) = 2j + \frac{q}{2} + 1$. The final receiver of m_j in sequence #1 is $\min\left(p + q - 1, j + p + \frac{q}{2}\right)$. If the final receiver of m_j in sequence #1 is $p + q - 1$, the case is finished. Otherwise the final receiver is $j + p + \frac{q}{2}$. Since $j \in \left[\frac{p}{2}, p - 1\right]$, $j + p + \frac{q}{2} \geq 2j + 1 + \frac{q}{2}$. Hence m_j in both sequences must have intersected. It follows from Theorem IV.3 that all receivers have received messages $\{m_j : j \in \left[\frac{p}{2}, p - 1\right]\}$.

V. Experimental Results

We implement the proposed algorithms for inter-group all-to-all broadcast on fully connected topology using MPI_Send, MPI_Recv, and MPI_Allgather with intra-group communicators as building blocks. To make a fair comparison, we emulate the root gathering algorithm using the MPI_Bcast and MPI_Allgather with intra-group communicators as building blocks. The timing performance of this implementation for the root gathering differs from the performance of calling MPI_Allgather with an inter-group communicator by less than 3%. Thus, comparing the proposed algorithms with the emulation of the root gathering algorithm is also a direct comparison. The root gathering algorithm is designed for topologies

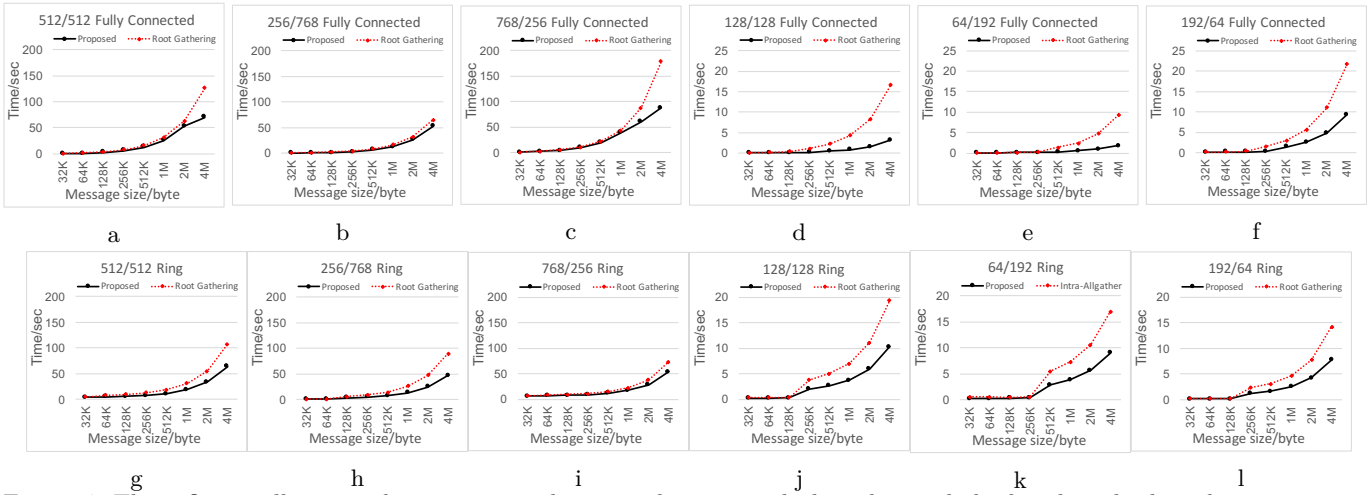

Figure 4: These figures illustrate the comparison between the proposed algorithm and the benchmark algorithm in terms of communication cost. The title format is "no. senders/no. receivers, topology". The unit of horizontal axis is number of bytes per message. The unit of vertical axis is the communication cost in seconds.

that supports binary indexing, so it is highly inefficient on a ring topology. Thus, we use the intra-group Allgather as the benchmark for the ring topology. We implement the proposed algorithms using MPI_Sendrecv function under ring topological constraints. This implementation is compared with the intra-group Allgather algorithm under ring topological constraints implemented using the same MPI_Sendrecv function. Improvement of communication cost is computed as the benchmark communication cost divided by the proposed algorithm communication cost.

The experiments are performed on Cori, a Cray XC40 supercomputer at the National Energy Research Scientific Computing Center (NERSC). We use the Cray MPI compiler (cray-mpich/7.6.0) for MPI functions, which is based on MPICH. We adopt two process configurations. The first configuration has 256 nodes. Every node is assigned with a single process. For hybrid MPI and OpenMP applications that use shared memory at the same node and process communications among nodes, this setup is reasonable. The second configuration has 128 nodes. Every node is assigned with 8 processes. MPI programs that utilize cores spanning across nodes. The messages are randomly generated data with size from 32KB to 4MB. If message size smaller than 32KB is used, the bandwidth is trivial, so the proposed algorithm does not improve performance significantly. All timing results are averaged over eight trials of experiments.

A. Fully Connected Topology

Figure 4a illustrates the comparisons of communication cost with 512 senders and 512 receivers running on two groups of 64 compute nodes, respectively. The communication cost improvement is 1.8 with 4MB message size. Figure 4d illustrates the comparisons of communication cost with 128 senders and 128 receivers running on two groups of 128 compute nodes, respectively. The communication cost improvement is 5.3 with 4MB message size. As the message size increases, the difference of total communication cost becomes significant because the bandwidth starts to dominate the total communication cost. Moreover, the improvement increases as the number of processes becomes larger, since the total message size received by a process increases.

Figure 4b illustrates the comparisons of communication cost with 256 senders and 768 receivers running on two groups of 32 and 96 compute nodes, respectively. The communication cost improvement is 2.0 with 4MB message size. Figure 4e illustrates the comparisons of communication cost with 64 senders and 192 receivers running on two groups of 64 and 192 compute nodes, respectively. The communication cost improvement is 5.1 with 4MB message size.

Figure 4c illustrates the comparisons of communication cost with 768 senders and 256 receivers running on two groups of 96 and 32 compute nodes, respectively. The communication cost improvement is 1.2 with 4MB message size. Figure 4f illustrates the comparisons of communication cost with 192 senders and 64 receivers running on two groups of 192 and 64 compute nodes, respectively. The communication cost improvement is 4.4 with 4MB message size.

The communication cost improvements are much better for the configuration that assign one process per node. When multiple processes are assigned to the same node, the communication cost between process are not uniform, since a processes within the same node communicate with each other faster than processes located in remote nodes. This configuration violates the assumption we made in section 2. Nevertheless, we demonstrate that the proposed algorithm still run faster than the root gathering algorithm.

B. Ring Topology

Figure 4g illustrates the comparisons of communication cost with 512 senders and 512 receivers running on

two groups of 64 compute nodes, respectively. Figure 4j illustrates the comparisons of communication cost with 128 senders and 128 receivers running on two groups of 128 compute nodes, respectively. The communication cost improvements are 1.7 and 1.9 with 4MB message size.

Figure 4h illustrates the comparisons of communication cost with 256 senders and 768 receivers running on two groups of 8 and 24 compute nodes, respectively. Figure 4k illustrates the comparisons of communication cost with 64 senders and 192 receivers running on two groups of 2 and 6 compute nodes, respectively. The communication cost improvements are both 1.9 with 4MB message size. Compared with $p = q$, the improvements with large message size is larger. This improvement matches the complexity comparison in Table II because the difference of communication cost between the intra-group Allgather and the proposed algorithm is proportional to the number of receivers.

Figure 4i illustrates the comparisons of communication cost with 768 senders and 256 receivers running on two groups of 24 and 8 compute nodes, respectively. Figure 4l illustrates the comparisons of communication cost with 192 senders and 64 receivers running on two groups of 6 and 2 compute nodes, respectively. The communication cost improvements are 1.8 and 1.4 with 4MB message size. The communication cost improvement is less compared with the previous two cases, because the difference of communication cost between the intra-group Allgather and the proposed algorithm is positively related to the number of receivers in Table II.

The theoretical communication cost improvements for the proposed ring algorithms are 1.5, 1.14, and 1.6 for $p = q$, $p = 2q$, and $2p = q$ when there are p senders and q receivers according to Table II. We expect to observe communication cost improvement close to the theoretical values when the total number of processes is large enough. This observation can be made as we increase the total number of processes from 256 to 1024.

VI. Conclusions

Inter-group communication is a fundamental communication pattern for scientific workflow systems. In this paper, we propose optimal algorithms of inter-group half-duplex all-to-all broadcast designed for both ring topology and fully connected topologies. Both theoretical and experimental results have shown that the proposed algorithms outperform the benchmarks. In the future, it is possible to extend our proposed algorithms from ring topology to torus topology.

VII. Acknowledgment

This work is supported in part by the following grants: NSF awards CCF-1409601, DOE awards DE-SC0007456 and DE-SC0014330. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] MPI Forum, *MPI: A Message-Passing Interface Standard. Version 3.1*, June 4th 2015, www.mpi-forum.org.

[2] Q. Kang, J. L. Träff, R. Al-Bahrani, A. Agrawal, A. Choudhary, and W.-k. Liao, "Full-duplex inter-group all-to-all broadcast algorithms with optimal bandwidth," in *Proceedings of the 25th European MPI Users' Group Meeting*, ser. EuroMPI'18. New York, NY, USA: ACM, 2018, pp. 1:1–1:10. [Online]. Available: http://doi.acm.org/10.1145/3236367.3236374

[3] J. Liao, B. Gerofi, G.-Y. Lien, T. Miyoshi, S. Nishizawa, H. Tomita, W.-K. Liao, A. Choudhary, and Y. Ishikawa, "A flexible i/o arbitration framework for netcdf-based big data processing workflows on high-end supercomputers," *Concurrency and Computation: Practice and Experience*, 2017.

[4] J. Hacker and W. Angevine, "Ensemble data assimilation to characterize surface-layer errors in numerical weather prediction models," *Monthly Weather Review*, vol. 141, no. 6, pp. 1804–1821, 2013.

[5] S. Habib, A. Pope, H. Finkel, N. Frontiere, K. Heitmann, D. Daniel, P. Fasel, V. Morozov, G. Zagaris, T. Peterka *et al.*, "Hacc: Simulating sky surveys on state-of-the-art supercomputing architectures," *New Astronomy*, vol. 42, pp. 49–65, 2016.

[6] C. Sewell, K. Heitmann, H. Finkel, G. Zagaris, S. T. Parete-Koon, P. K. Fasel, A. Pope, N. Frontiere, L.-t. Lo, B. Messer *et al.*, "Large-scale compute-intensive analysis via a combined in-situ and co-scheduling workflow approach," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 50.

[7] W. Gropp and E. Lusk, "Fault tolerance in message passing interface programs," *International Journal of High Performance Computing Applications*, vol. 18, no. 3, pp. 363–372, 2004.

[8] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi, "Enabling in-situ execution of coupled scientific workflow on multi-core platform," in *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*. IEEE, 2012, pp. 1352–1363.

[9] C. Docan, F. Zhang, T. Jin, H. Bui, Q. Sun, J. Cummings, N. Podhorszki, S. Klasky, and M. Parashar, "Activespaces: Exploring dynamic code deployment for extreme scale data processing," *Concurrency and Computation: practice and Experience*, vol. 27, no. 14, pp. 3724–3745, 2015.

[10] P. Silva and J. Silva, "Implementing mpi-2 extended collective operations," *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp. 681–681, 1999.

[11] A. Skjellum, N. E. Doss, and K. Viswanathan, "Intercommunicator extensions to mpi in the mpix (mpi extension) library," Technical Report MSU-940722, Mississippi State University—Dept. of Computer Science, Tech. Rep., 1994.

[12] "Mpich3," http://www.mpich.org/downloads/, 2017.

[13] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[14] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: design and analysis of algorithms*. Benjamin/Cummings Redwood City, 1994, vol. 400.

[15] E. Chan, M. Heimlich, A. Purkayastha, and R. Van De Geijn, "Collective communication: theory, practice, and experience," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 13, pp. 1749–1783, 2007.

[16] D. P. Bertsekas, C. Özveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, "Optimal communication algorithms for hypercubes," *Journal of Parallel and Distributed Computing*, vol. 11, no. 4, pp. 263–275, 1991.

[17] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in mpich," *The International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 49–66, 2005.

[18] J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby, "Efficient algorithms for all-to-all communications in multiport message-passing systems," *IEEE Transactions on parallel and distributed systems*, vol. 8, no. 11, pp. 1143–1156, 1997.

[19] A. Bar-Noy and S. Kipnis, "Broadcasting multiple messages in simultaneous send/receive systems," *Discrete Applied Mathematics*, vol. 55, no. 2, pp. 95–105, 1994.