CHAPTER 3

EFFECTIVE BROADCASTING IN PARALLEL NETWORKS USING NETWORK CODING

This chapter describes the applicability of linear network coding on parallel architecture for multi-source finite acyclic network. Further it discloses the problem in which different messages in diverse time periods are broadcast and every non-source node in the network decodes and encodes the message based on further communication. It is shown how to minimize the communication steps and time complexity involved in transfer of data from node-to-node during parallel communication. Multi-Mesh of Trees (MMT) topology is used for implementing network coding. To evaluate results with and without network coding, All-to-All Broadcast algorithm is considered. For data communication at different step, the source and destination nodes changes according to the flow of data in the algorithm.

3.1 Network Coding and Parallel Communication

Parallel architectures involve parallel communication with the aim of receiving complete information at higher rate. To reduce the complexity of parallel communication we have considered Multi-Mesh of Trees (MMT) [115] architecture and implemented Linear Network Coding (LNC) [2]. We have proposed Linear-Code Multicast with Parallel Architectures (LCM-PA) which achieves max-flow from the source of each receiving nodes. While achieving the desired throughput in a multicast scenario, we have minimized the communication steps and time complexity. We have developed a novel application of LNC for parallel network communication.

The meaning of parallel algorithm is to reduce the time complexity of several problems to enable fast and efficient communication. Several interconnection networks have developed parallel processing approaches based on the mesh topology. We have presented an efficient application of LNC [110-114], for MMT parallel network. This approach asymptotically achieves the capacity of multicast networks with network coding [2]. We assume a parallel
multisource multicast, which is possible with correlated sources on MMT architecture. We have used an approach in which all nodes other than the receiving nodes perform random linear mapping from inputs on outputs (see figure 1).

In each incoming transmission from source, the destination has knowledge of overall linear combination of data set. This information updates at each coding node by applying the same linear mapping to the coefficient vectors as applied to the information signals. As an example, let the set of two bits \((d_1, d_2)\) multicast in a directed parallel network \((N)\). Figure 1 shows that from the source node \(P_1\) (unique node, without any incoming at that instant of time) to node \(P_2, P_3\) and \(P_4, P_7\).

![Figure 3.1](image)

**Figure 3.1:** A network \((N)\) used, as an example, to explain LNC with coefficient added at each data transfer from different nodes (the network has seven nodes \(P_1, P_2... P_7\) and nine edges \(P_1 P_2, P_1 P_5, P_2 P_3, P_2 P_3, P_3 P_6, P_4 P_5, P_6 P_7, P_6 P_7\) directed in this order). \((d_1, d_2)\) is the set of data being multicast to destinations, and coefficients \(\zeta_1, \zeta_2... \zeta_6\) are randomly chosen elements of a finite field. Each link represents the data transmission.

Figure 3.1 shows information multicast in a network \((N)\) with network coding. So, let us explain network coding using network \((N)\) given in figure 1. Node \(P_1\) aims to multicast the data set \((d_1, d_2)\) to destination nodes \(P_3\) and \(P_7\). A randomly chosen coefficient of finite field clubs the data at each node and then transferred. The transmission continues as: \((d_1, d_2)\) sent to both \(P_2\) and \(P_4\) with different coefficients \((\zeta_1, \zeta_2\) and \(\zeta_3, \zeta_4)\). Node \(P_2\) receives \(\zeta_1d_1+\zeta_2d_2\) and \(P_4\) receives \(\zeta_3d_1+\zeta_4d_2\). Then both \(P_2\) and \(P_4\) transfer the information to \(P_5\). Node \(P_5\) and \(P_6\) sends \(\zeta_5(\zeta_1d_1+\zeta_2d_2)+\zeta_6(\zeta_3d_1+\zeta_4d_2)\) to \(P_3\) and \(P_7\), which decodes to receive the data \((d_1, d_2)\).
This approach shows that multicast of different data is possible using network coding in the network. It is right to say that flow of information and flow of physical commodities are two different things [2]. So to code information does not increase its content. To transmit information from the source to destination, capacity of a network can become higher [2]. We have implemented LNC on MMT and it results in novel efficient way of data communication by coding information to multicast on other processors. This approach has reduced the chance of error and increased the capacity to transmit data between processors. The linearity of code, for parallel transmission, makes encoding (at source end) and decoding (at receiving end) easy to implement in practice.

3.2 AAB (All-to-All Broadcast) on Parallel Network

We have implemented AAB on parallel network (MMT). For implementation, we have used AAB algorithm, which involves ten steps of algorithms to transfer and receive information [116]. We considered the MMT network with \( n = 8 \), where \( n \) is the number of processors and we considered \( N = n^2 \times n^2 \), \( \forall n \in N \) and a block = \( n \times n = \text{row } \times \text{column} \). MMT is better than other traditional parallel networks (e.g., Multi-Mesh (MM) [117, 118]) based on the topological properties of MMT comparable based on efficiency parameters. Figure 3.2 shows a comparison of these networks based on some parameters and figure 3.3 shows comparison between 2D Sort on MM and MMT for different number of processor. Table 1 [115, 119] shows characteristics of various processor organizations based on the network optimization parameters. This shows that MMT is more efficient then these network architectures.

![Graph showing comparison between MMT and MM](image)

**Figure 3.2:** Comparison of MMT and MM on the basis of Communication links, Solution of Polynomial Equations, One to All and Row & Column Broadcast.
CHAPTER 3

Figure 3.3: A comparison between 2D Sort on MM and MMT for different values of processor.

Table 3.1: Characteristics of Various Processor Organizations.

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Diameter</th>
<th>Bisection Width</th>
<th>Constant Number of Edges</th>
<th>Constant Edge Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-D mesh</td>
<td>$n$</td>
<td>$n - 1$</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2-D mesh</td>
<td>$n^2$</td>
<td>$2(n - 1)$</td>
<td>$n$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3-D mesh</td>
<td>$n^3$</td>
<td>$3(n - 1)$</td>
<td>$n^2$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Binary tree</td>
<td>$2^n - 1$</td>
<td>$2(n - 1)$</td>
<td>1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>4_ary hypertree</td>
<td>$2^n(2^n - 1)$</td>
<td>$2n$</td>
<td>$2^{n+1}$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Pyramid</td>
<td>$4n^2 - 1)/3$</td>
<td>$2\log n$</td>
<td>$2n$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Butterfly</td>
<td>$(n + 1)2^n$</td>
<td>$2n$</td>
<td>$2^n$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Hypercube</td>
<td>$2^n$</td>
<td>$n$</td>
<td>$2^{n-1}$</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Cube-connected cycles</td>
<td>$n2^n$</td>
<td>$2n$</td>
<td>$2^{n-1}$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Shuffle-exchange</td>
<td>$2^n$</td>
<td>$2n - 1$</td>
<td>$\geq 2^{n-1}/n$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>De Bruijn</td>
<td>$2^n$</td>
<td>$n$</td>
<td>$2^n/n$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MMT</td>
<td>$n^4$</td>
<td>$4\log n + 2$</td>
<td>$2(n - 1)$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MM</td>
<td>$n^4$</td>
<td>$2n$</td>
<td>$2(n - 1)$</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Now, to explain the algorithm, we consider $N = 8^2 \times 8^2 = 4096$ nodes, as the size of network, where each block consists of $8 \times 8$ i.e., row $\times$ column. To explain each step of algorithm we have used either one row or column to show the data flow. For every step the data flow varies so each step need different algorithm. Figure 3.4 shows first row in first block of the network and the connectivity between the processors based on the topological properties of MMT [115]. We have considered that each processor is having a Working Array (WA) which consist of the processor index ($P_i$) and information associated with that processor ($I_i$). The size of working array depends on the size of network used, i.e. for $n = 8$, the size of WA = 8.
**Figure 3.4:** Shows initial condition of processors containing WA (only one row of a block of 8 × 8 MMT is shown)

The figure 3.5 (a) shows the position of data after completion of step 1 and figure 3.5 (b) shows the content of WA1 after step 1.

**Algorithm 1. Step 1 of AAB**

- a. /* This operation is common between all processors of each row of each block,
- b. Each node is represented by \( n_{(a, b, i, j)} \); where \( a, b \) are the block index and \( i, j \) are node index
- c. The transfer is conducted in order \( \frac{n_{(a, b, i, j)}}{2 \times \text{count of iteration} - 1} < j \leq \frac{n_{(a, b, i, j)}}{2 \times \text{count of iteration} - 1} */

1: Starting from each row of each block of network, such that the processor with greater index value will transfer data to lower index processors linked according to the topological properties of network.
2: repeat
3: Select nodes \( n_{(a, b, i, j)} \) from each block of network such that at each transfer the block is divided in two parts (e.g. if \( N = 40 \), number of nodes in blocks will also be 40 and division will be 1 to 20 and 21 to 40th index position) and transfer message to remaining nodes \( n_{(a, b, i, j)} \) linked according to topological properties of this network.
4: Select nodes \( n_{(a, b, i, j)} \) from each block of network such that at each transfer these nodes are divided in two parts (same as in 3; i.e. \( n_{(a, b, i, j)} \) and \( n_{(a, b, i, j)} \)) and \( n_{(a, b, i, j)} \) linked according to topological properties of this network.
5: until all nodes have finished transmitting and forwarding.

**Algorithm 2. Step 2 of AAB**

- a. /* This operation is common between all root processors of each row of each block,
- b. Root processors of each row of a block are identified as in figure 3.5 (a),
- c. The transfer of information of all root processors of respective rows is conducted according to connectivity. */

1: Starting from each row of each block. The root nodes of respective rows will transfer data to
connected nodes of that row.
2: repeat
3: until all nodes have received the information of root processors.

After the completion of step 2 the position of data in a row is shown in figure 3.6. The data from the root node of a row of all blocks of network receives the complete information of that row as the content of WA1.

Algorithm 3. Step 3 of AAB

1: Starting from each column of each block of network, such that the processor with greater index value will transfer data to lower index processors linked according to the topological properties of network.
2: repeat
3: Select nodes $n_{(a,b,i,\frac{N}{2}+1)}$, $n_{(a,b,i,\frac{N}{2}+2)}$, $n_{(a,b,i,\frac{N}{2}+3)}$, ..., $n_{(a,b,i,N)}$ ∈ $N$ from each block of network such that at each transfer the block is divided in two parts (e.g. if $N = 40$, number of nodes in blocks will also be 40 and division will be 1 to 20 and 21 to 40th index position) and transfer message to remaining nodes $n_{(a,b,i,1)}$, $n_{(a,b,i,2)}$, $n_{(a,b,i,3)}$, ..., $n_{(a,b,i,\frac{N}{2})}$ ∈ $N$ linked according to topological properties of this network.

Figure 3.6: After Step 2

Figure 3.7: a) Step 3     b) After Step 3     c) Step 4      d) After Step 4

Figure 3.7: shows the Step 3 and 4 in which the communication is performed in each column of each block of the network. After the completion of step 4 each column of each block of network consists of complete information of respective column.
Algorithm 4. Step 4 of AAB

1. Starting from each column of each block. The root nodes of respective columns will transfer data to connected nodes of that row.
2. repeat
3. until all nodes have received the information of root processors.

Algorithm 5. Step 5 of AAB (Interblock Communication)

1. Starting from each blocks of each rows the information is communicated to the root processors of respective block in such a manner that the processor index \( n(a,\beta,i,j,N) \).
2. In one communication step this information is broadcasted to every root processor of respective block of respective row. This step is performed on entire network.
Note: At the end of this step every root processor contains the information of complete block from which this information is broadcasted.

Algorithm 6. Step 6 of AAB (Interblock Communication)

1. Starting from each column of each block of network, such that the processor with greater index value will transfer data to lower index processors linked according to the topological properties of network.
2. repeat
3. Select nodes \( n(a,\beta,i,j,N+1) \), \( n(a,\beta,i,j,N+2) \), \( n(a,\beta,i,j,N+3) \) \( \ldots \) \( n(a,\beta,i,j,N) \in N \) from each block of network such that at each transfer the block is divided in two parts and transfer message to remaining nodes \( n(a,\beta,i,1) \), \( n(a,\beta,i,2) \), \( n(a,\beta,i,3) \) \( \ldots \) \( n(a,\beta,i,N) \in N \) linked according to topological properties of this network.

Algorithm 7. Step 7 of AAB (Interblock Communication)

/* One-to-all broadcast is used in the block*/

To transfer the information of a block in a row to other block of respective rows the one-to-all broadcast algorithm is used.
Note: At the end of this step, complete blocks of each row have information of all processors in that row.

Algorithm 8. Step 8 of AAB (Interblock Communication)

/* The step is performed using the horizontal interblock links of this network which transfers the information of all the blocks of respective rows to the root processors of respective block with processor index \( i = N \)*/

1. Starting from each blocks of each columns the information is communicated to the root processors of
respective block in such a manner that the processor index $n_{(a, b, i = N, j)}$.

2: In one communication step this information is broadcasted to every root processor of respective block of respective column. This step is performed on entire network.

*Note: At the end of this step every root processor contains the information of complete block from which this information is broadcasted.*

Algorithm 9. Step 9 of AAB

| a. */ This operation is common between all processors of each row of each block, |
| b. Each node is represented by $n_{(a, b, i)}$; |
| c. The transfer is conducted in order $n_{(a, b, i)} < i \leq n_{(a, b, i)}$ */ |

1: Starting from each row of each block of network, such that the processor with greater index value will transfer data to lower index processors linked according to the topological properties of network.

2: repeat

3: Select nodes $n_{(a, b, i, 1)}^N, n_{(a, b, i, 2)}^N, ... n_{(a, b, i, N)}^N \in N$ from each block of network such that at each transfer the block is divided in two parts and transfer message to remaining nodes $n_{(a, b, i, 1, 1)}^N, n_{(a, b, i, 2, 2)}^N, ... n_{(a, b, i, N)}^N \in N$ linked according to topological properties of this network.

*Note: The message will be transferred from higher processor index to lower.*

4: Select nodes $n_{(a, b, i, 1)}^N, n_{(a, b, i, 2)}^N, ... n_{(a, b, i, N)}^N \in N$ (other than the nodes from which message has already transferred) from each block of network such that at each transfer these nodes are divided in two parts (same as in 3; i.e. $n_{(a, b, i, 1)}^N, n_{(a, b, i, 2)}^N, ... n_{(a, b, i, N)}^N$) and $n_{(a, b, i, 1, 1)}^N, n_{(a, b, i, 2, 2)}^N, ... n_{(a, b, i, N)}^N \in N$. Now $n_{(a, b, i, 1)}^N, n_{(a, b, i, 2)}^N, ... n_{(a, b, i, N)}^N$ will transfer respective messages to $n_{(a, b, i, 1)}^N, n_{(a, b, i, 2)}^N, ... n_{(a, b, i, N)}^N$, linked according to topological properties of this network.

5: until all nodes have finished transmitting and forwarding.

Algorithm 10. Step 10 of AAB

/* AAB is used in the block*/

Select block from each column to transfer information of a block in a column to other block of respective columns for this AAB is used.

*Note: At the end of this step, all the processors of each block contains information of all processors of the network.*

3.3 LNC on AAB using MMT

We have implemented LNC for each step to make the communication faster and increase the rate of information transmitted from each node. We considered network as delay-free (acyclic) and $o(l) \neq d(l)$. The algorithm results are analyzed later with $n = 8$ processors. For each step, independent and different algorithms are used and linear coding is implemented with each algorithm. According to algorithm 1, all processors transfers data with $n = 8$ and $count = 1$ to 2 i.e., $(8/(2\times1 - 1)) < j \leq (8/2\times1) = 8 < j \leq 4$. This means the processors $P_1, P_2, P_3$ and $P_4$ will receive data from $P_5, P_6, P_7$ and $P_8$, shown in figure 3.8.
CHAPTER 3        EFFICIENT BROADCAST IN P.N. USING N.C.

Figure 3.8: (a) Shows the indexing of processors with respect to nodes in the figure. (b) Shows the direction of flow of data in step 1 of AAB algorithm on MMT, $P_1$, $P_2$, $P_3$ and $P_4$ are the processor receiving data and $P_5$, $P_6$, $P_7$ and $P_8$ are the sending processors. The dotted line distinguishes between the receiving and sending processors in first iteration of step 1.

**Step 1:** Linear coding is implemented on $P_1$, $P_2$ and $P_3$ processors, as these are receiving a set of data form source processors $P_5$, $P_6$, $P_7$ and $P_8$ in first iteration. Processor $P_8$ is source and $P_4$ is its destination; $P_7$ and $P_6$ are sources and $P_3$ is their destination; lastly in $\log n$ iteration i.e. (3 iteration for $n = 8$), $P_1$ will receive data from $P_2$ and $P_3$. After implementation of LNC according to LCM-PA on these sources and destinations, step 1 will work as in figure 3.9. During first iteration of AAB on MMT, LCM-PA will work as in figure 3.9 (a). Nodes $P_5$, $P_6$, $P_7$ and $P_8$ send data to $P_2$, $P_3$, $P_2$ and $P_4$ respectively. So, the complete set of data from all processors reached processor $P_1$, i.e. after execution of step 1 all data in a row will reach its root processors. But due to LCM-PA the data reached $P_1$ will have time complexity of $(\log n - 1)$, as one step is reduced during transfer of the data using LCM-PA model.

**Step 2:** The root processors of each row, ($P_1$: root processor of first block and first row) will broadcast the data (from $P_1$: $\zeta_1 d_8 + \zeta_2 d_7 + \zeta_3 d_6 + \zeta_4 d_5 + \zeta_5 d_4 + \zeta_6 d_3 + \zeta_7 d_2$) to all the processors of respective row using intrablock links transfer, see figure 3.10.

The time complexity for this step will be reduced by $n$ i.e., $n(\log n-1)$. This step is a broadcasting step in each block with intrablock links of MMT. At the end of this step, complete data from root processor is received by other processors of that row. LCM-PA is applied at the same level as in step 1, but the size of data increases to $n$.

**Step 3:** This step is similar to step 1, but in this step the data is broadcasted in column-wise order of each block. Linear coding is implemented on $P_{11}$, $P_{12}$ and $P_{13}$ processors, as these are receiving a set of data form source processors $P_{15}$, $P_{16}$, $P_{17}$ and $P_{18}$ in first iteration. Processor $P_{18}$ is source and $P_{14}$ is its destination; $P_{17}$ and $P_{16}$ are sources and $P_{13}$ is their destination;
lastly in log \( n \) iteration i.e. (3 iteration for \( n = 8 \)), \( P_{11} \) will receive data from \( P_{12} \) and \( P_{13} \). After implementation LCM-PA on these sources and destinations, step 3 works as in figure 3.11.

**Step 4:** In this step all the root processors of each column and each block, (\( P_{11} \): root processor of first block and first column) will broadcast the data (from \( P_{11} \): \( \zeta_1 d_{18} + \zeta_2 d_{17} + \zeta_3 d_{16} + \zeta_4 d_{15} + \zeta_5 d_{14} + \zeta_6 d_{13} + \zeta_7 d_{12} \)) to all the processors of respective column using intrablock links transfer, see figure 3.12. The time complexity of this step is reduced by \( n^2 \) i.e. \( n^2 (\log n - 1) = n^2 \log n - n^2 \). The coefficient value (\( \zeta_i \)) in step 4 is different from the coefficient value in step 1.

**Figure 3.9:** (a) Iteration first of step 1; data from processors \( P_3 \), \( P_6 \), \( P_7 \) and \( P_8 \) is sent to processors to \( P_2 \), \( P_3 \), \( P_5 \) and \( P_4 \) respectively. (b) Iteration second of step 1; data from processors \( P_4 \) and \( P_1 \) is sent to processors to \( P_2 \) and \( P_4 \) respectively. (c) Iteration third of step 1; data from processors \( P_5 \) and \( P_3 \) is sent to processors \( P_1 \).

**Figure 3.10:** The data from each row root processor is broadcasted to other processors of respective row in each block.
Step 5: After step 4, each processors of respective columns contains information of all processors of that column. Step 5 perform the interblock communication using the horizontal interblock links which transfers this information (of all the blocks of respective rows) to the root processors (of respective block) and this requires one communication step (CS). The time complexity of this step will be same as of AAB i.e. 1CS.

Step 6: Using step 3, for transferring information of all the processors in the column at the processors with P_ID \((j=n)\), so the WA of all the processors is transferred in the column in the order \(n/(2\text{count}−1) < j ≤ n/(2\text{count})\). Time Complexity of step 6: \(n^3\log n\).

Step 7: Call one–to–all algorithm in the block to transfer the INFO of other blocks (of respective rows) in \(n^3\log n\) time. At the end of this step, complete blocks of each row have INFO of all the processors in that row. Time Complexity of step 7: \(n^3\log n\).

**Figure 3.11:** (a) Iteration first of step 3; data from processors \(P_{13}, P_{16}, P_{17}\) and \(P_{18}\) is sent to processors to \(P_{14}\). \(P_{13}, P_{16}\) and \(P_{17}\) respectively. (b) Iteration second of step 3; data from processors \(P_{14}\) and \(P_{13}\) is sent to processors to \(P_{12}\) and \(P_{11}\) respectively. (c) Iteration third of step 3; data from processors \(P_{12}\) is sent to processors \(P_{11}\).

Step 8: This step performs the interblock communication using horizontal link transfer that transfers the INFO (of all the blocks of respective column) to the root processors (of respective block) with P_ID \((i=n)\), and this requires one communication step. Time Complexity of step 8: 1CS.
CHAPTER 3

Step 9: Using step 1 transfer of INFO of all the processors with P_ID (i=n). Time Complexity of step 9: $n^4\log n$.

Step 10: Call AAB algorithm in the block to transfer the INFO of other blocks that column in the block with $n^4\log n$ time complexity. Time Complexity of step 10: $n^4\log n$. At the end of this step all the processors of each block have the INFO of all processors of other blocks.

Figure 3.12: The data from each column root processor is broadcasted to other processors of respective column in each block.

3.4 Results and Simulations

To implement linear coding using AAB on MMT enables sharing of data between multiple processors more convenient and easy. As the algorithm becomes complex, the involvement of processors also increases. For parallel architectures, important issue is to make these architectures more processor utilitarian, otherwise the processors in these architectures are idle, and all are not in use at every step of algorithms. Higher coefficient participates to broadcast data compared with LCM-PA. So, this makes the algorithm less complex as fewer amounts of coefficients involves in broadcasting data using linear coding. While broadcasting the data in AAB, the time involved to communicate and deliver/receive data from different
processors is more. The fall of time complexity at different number of processors makes positive implementation of algorithm.

The algorithm starts with executing each step in the order defined as step 1...step 10. As it starts, the involvement of each processor also increases. In parallel processing the algorithm starts with active processor and involves other processors as it progresses [109]. Figure 3.13 show participation of processors with average percentage of iteration in each step.

![Graph showing involvement of processors at different steps of algorithm.](image)

**Figure 3.13:** Involvement of processors at different steps of algorithm.

Based on result in figure 3.13, we conclude that as the iterations increase the involvement of processors also increases. The algorithm with LCM-PA approach utilizes the maximum number of processors besides without LCM-PA approach. So to utilize processors in parallel architectures also increases using linear coding.

### 3.5 Chapter Outline

This chapter presents LCM-PA: a model of linear coding on parallel architecture. This proposed model efficiently implements AAB algorithm over MMT network. The results are discussed with comparative time complexity after implementation with LCM-PA. Further it
is shown that the model is network independent and can be implemented on any parallel architecture considering the topological properties of that architecture.