EFFICIENT MULTICAST ALGORITHMS FOR MESH AND TORUS NETWORKS

ANKIT MALANI

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE AT

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

DECEMBER 2012

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Ankit Malani

Entitled: Efficient Multicast Algorithms for Mesh and Torus Networks

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. N. Tsantalis	Chair
Dr. J. W. Atwood	Examiner
Dr. D. Goswami	Examiner
Dr. H. Harutyunyan	Supervisor

Approved by ______________________________
Chair of Department or Graduate Program Director

____
Dean of Faculty

Date ______________________________

ABSTRACT

Efficient Multicast Algorithms for Mesh and Torus Networks

Ankit Malani

With the increasing popularity of multicomputers, efficient way of communication within its processors has become a popular area of research. Multicomputers refer to a computer system that has multiple processors, they have high computational power and they can perform multiple tasks concurrently. *Mesh* and *Torus* are some of the commonly used network topologies in building multicomputer systems. Their performance highly depends on the underlying network communication such as multicast. Multicast is a communication method in which a message is sent from a source node to a certain number of destinations. Two major parameters used to evaluate multicast are *time* that a multicast process takes to deliver the message to all destinations and *traffic* that indicates the number of links used for this process. Research indicates that in general, it is NP-complete to find an optimal multicasting algorithm which is efficient on both time and traffic.

This thesis suggests two new algorithms to achieve multicast in mesh and torus networks. Extensive simulations of these algorithms show that in practice they perform better than existing ones.

ACKNOWLEDGEMENT

This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this work.

First and foremost, I would like to thank my supervisor Dr. Hovhannes A. Harutyunyan, who has given me help and support throughout my studies at Concordia University. I have learnt a lot from him in both professional and personal life. He has been an incredible mentor to me.

I would like to thank my parents Mrs. Usha and Mr. Ramesh Chand Malani and my sister Ms. Namrata Malani for their love and encouragement since my childhood. This work would not have been possible without their support and patience.

Last but not the least, I would like to express my deep gratitude to the faculty, staff and fellow graduate students at Concordia University for their assistance.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

There has been an immense development in the field of hardware over the past decade. The technology used in computers has changed in leaps and bounds. However as compared to the growth of hardware, the growth of software has been relaxed. Few software use the computational power of a multi-processor computer system, hence do not exploit the power of underlying system to the expected magnitude.

Many fields such as telephone networks, aircraft control systems, etc., need enormous computational capabilities. As more and more people are getting connected to Internet, the demand of (processed) contents such as websites, video is also increasing. *Multicomputers* - a computer system that has multiple processors is used to meet these demands [8]. In these systems, each processor is connected to other processors in the same system. The computational power of multicomputers is high and they can perform multiple tasks concurrently. These systems are also known as Massively Parallel Computers (MPC) [16]. Each processor has its own local memory and does not share it with other processors. For the purpose of communication among them, messages have to be passed within the network of processors. The overall performance of a system can be improved by passing the message in an efficient manner. Messages can be passed using unicast (one to one), multicast (one to many) or broadcast (one to all) method. Unicast and broadcast are special cases of multicast [4]. Better the multicast system better is the performance of the network. Thus, highly efficient approaches are required to pass the message.

The performance of a multicast routing approach can be evaluated by two main factors: *time* and *traffic*. *Traffic* signifies the number of links used in distributing the message from source to all destinations. *Time* indicates the maximum time units used to complete the message distribution from the source node to any destination node. A routing approach that reduces time and traffic would be an efficient one. *Mesh* and *Torus* are two of the network topologies used to construct a multicomputer system, because of their important properties such as low cost (number of links), scalability and ease of implementation [9, 16]. Many algorithms have been developed recently for mesh-connected multicomputer systems [8, 13, 20, 32]. Several of these are based on unicast, which is not an efficient way to pass a message as it leads to high multicast time and traffic. However there are few algorithms, which use tree-based multicast approach and have better performance as compared to the unicast approach [3, 6, 12, 31].

In this research, two new tree-based multicast algorithms have been devised, which in practice perform better than earlier developed algorithms.

1.2 Problem Description

Multicasting is a technique by which a message or information is delivered to a group of destination processors or computers in a network [9]. Multicasting can be applied to many real world applications such as push media, monitoring, concurrent processing and many more.

A network is a collection of hardware components, such as processors, routers, etc., arranged in a manner so that they can communicate to each other through a direct or indirect connection.

Network topologies, switching technology and routing algorithms [16, 20] are some of the factors that play a vital role in the efficiency of a multicomputer system

Why is multicast required?

The main purpose of multicast is to improve the performance of the underlying network. Multicast is used for *file distribution and caching* in distributed and parallel systems to transfer big chunks of data. It is useful in *monitoring* a system and updating the results, such as stock prices and security. It can also be used to support concurrent data processing in parallel algorithms and for the coordination within a multicomputer system.

Performance of a multicomputer system depends not only on processor's speed, but also on the underlying communication system. The main factor that affects the performance of communication system is communication latency [17]. Communication latency is defined as the maximum time taken by a message to travel from the source node to any destination node. Therefore, it is always desired to have communication latency efficient routing algorithms for multicomputers.

Implementation Techniques

There are several ways by which multicast can be achieved. The first and foremost approach is to route an individual packet to every destination, which is formally known as unicast-based multicast [13]. Another approach is to use broadcast in which every node passes the message to its neighboring node until all the nodes in the system gets the same copy of the message. The problem with both of these approaches is that they generate a high volume of traffic.

A better way to get multicast done in an efficient manner is to use tree-based multicast [11, 18, 20, 21]. In this approach, message from the source node is distributed to multiple neighboring nodes; these nodes then forward the message to other nodes, which are located at a farther distance from the source node. This method helps to reduce the number of links in the network that is used to achieve multicast. This approach also overcomes the disadvantages of other methods and is highly capable to improve the overall performance of the system.

Challenges in tree-based multicast

The main challenge in tree-based multicast approach is to find an ideal multicast tree that can transfer the message from source node to all destinations in an efficient manner.

As described earlier, *time* and *traffic* are two parameters by which the performance of a multicast algorithm is judged. These parameters are indirectly related to each other in a manner such that with a decrease in one parameter, the other parameter automatically increases. It has been theoretically proved that finding an optimal multicast tree, based on both parameters *time* and *traffic* in a mesh-connected network is *NP complete*.

1.3 Scope of Research

The main focus of this research study is on the efficiency of the multicast algorithm under a theoretical model. Many issues, such as deadlock, faulty link, etc., can hamper the performance of a network. In this study, an ideal scenario is assumed such that there are no deadlocks or faulty links. *2*-dimensional mesh and torus architectures are considered for this study. Algorithms are designed for a single message, which has to be delivered from a source node to a set of destination nodes available in the same network. Since it is an NP complete problem, the objective is to achieve a near optimal solution.

1.4 Organization of the Thesis

Rest of the thesis is organized as follows. Chapter 2 covers a detailed study of basics, previously conducted research and the latest work in this field. Chapter 3 presents two new multicasting algorithms. Chapter 4 covers extensive simulation and analysis of new algorithms with the existing ones and Chapter 5 provides conclusions and future work.

Chapter 2

Literature Review

2.1 Preliminary

Some fundamentals to understand multicast problem are covered in this section.

2.1.1 Basics of Network

2.1.1.1 Network

A network or a computer network, in a very broad manner, can be described as a collection of autonomous computers (also known as nodes) and other hardware devices interconnected by a single technology. Two computers are interconnected if they are able to exchange information between them. Networks can be classified into different categories based on their characteristics such as scale, topology, protocol used and many others. [22]

In computer science, a network can also be formed as a graph. A node in the network can be visualized as the vertex of the graph and a link connecting two nodes can be visualized as an edge of the graph. Hence, “node and vertex” and “link and edge” will be used interchangeably throughout this thesis.

2.1.1.2 Nodes

A node consists of a processor, local memory, a router and some input/output devices. Nodes in multicomputer systems communicate by passing messages through the underlying network. Hardware support is essential for the transmission of messages between nodes. A router is responsible to handle all the communication related tasks. If a node can send a message to at most one of its neighboring node at one time unit, the architecture is known as one-port architecture [15].

2.1.1.3 Network Topology

Network topology is the arrangement or placement of various nodes in a computer network [28]. It can be classified in two categories: direct and indirect.

A network in which there is an exclusive link between a node and each of its neighboring nodes is known as direct network. For example: hypercube, mesh.

However, in an indirect network, nodes are connected to other nodes through one or more switching elements. For example: bus, crossbar. These networks are easy to set up but the failure of switching element can disable the whole network. Also, the performance of a network reduces as the number of nodes increases.

Direct networks have emerged as a popular architecture for massively parallel computers, due to their properties such as scalability, fault tolerance and better performance [15]. Failure of a node will not result in the failure of the whole network and the processing capability of the overall network is directly proportional to the number of nodes present in the network.

Some common topologies are shown in *Figure 1*.

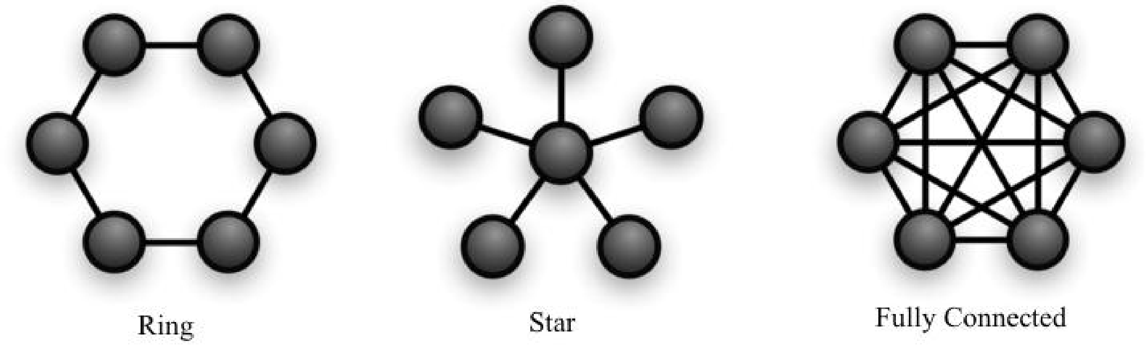

Figure 1 Different Network Topologies

If nodes are arranged in a *2*-dimensional array and are connected to their neighbors in both the dimensions, the arrangement is known as *2*-dimensional mesh, and can be formally defined as follows:

Definition 1 (*2-dimensional Mesh*): is an interconnection structure that has $k_1 \times k_2$ nodes, where k_1 and k_2 represents the number of nodes in dimension-*1* and dimension-*2* respectively. Each node in the mesh is identified by a *2*-coordinate vector (x_1, x_2), where $0 \leq x_1 \leq k_1 - 1$ and $0 \leq x_2 \leq k_2 - 1$. Two nodes (x_i, x_j) and (y_i, y_j) are connected if and only if there exists an i and a j such that $x_i = y_i \pm 1$ and $x_j = y_j$ or $x_j = y_j \pm 1$ and $x_i = y_i$ for all $j \neq i$.

Thus the number of neighbors for a node ranges from *2* to *4*, depending on its location in the mesh [15, 16].

In any mesh network, there is no connection between the first and the last node of each dimension, therefore a communication between these nodes needs to go through a long path. However, if a link is added between the first and the last node of each dimension in the network, the communication path between these nodes will be reduced significantly. Such mesh network with wrap around links is known as torus [10, 15, 16, 20].

Definition 2 (*2-dimensional Torus*): is similar to a 2-dimensional mesh in terms of interconnection structure of $k_1 \times k_2$ nodes, where k_1 and k_2 represents the number of nodes in dimension-*1* and dimension-*2* respectively. It differs from a *2*-dimensional mesh only at the interconnection structure. Two nodes (x_i, x_j) and (y_i, y_j) in torus are connected if and only if there exists an i and a j such that $x_i = (y_i \pm 1)\ mod\ k_1$ and $x_j = y_j$ or $x_j = (y_j \pm 1)\ mod\ k_2$ and $x_i = y_i$ for all $j \neq i$.

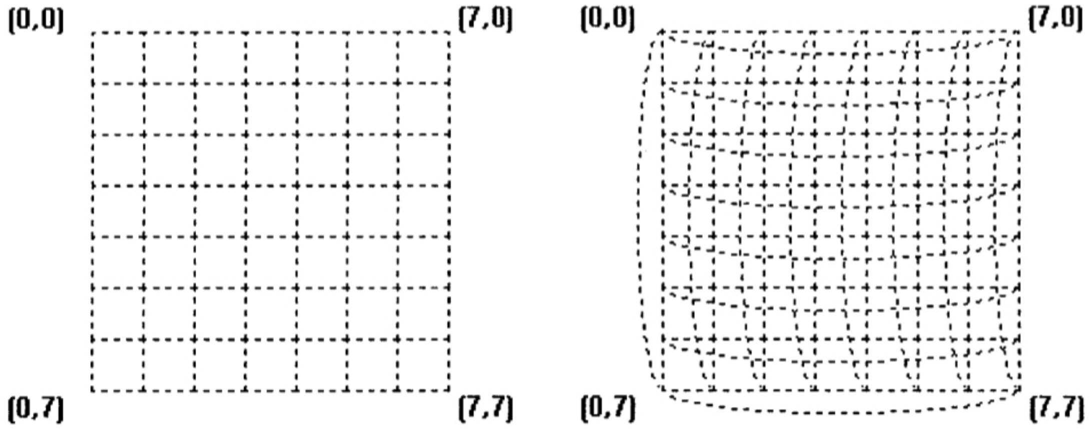

Figure 2 A 2-dimensional Mesh and Torus Network

As shown in *Figure 2*, torus is a symmetric topology in which the degree of a node is the same regardless of its location in the network. This is a very significant feature of torus as it is helpful in balancing the traffic load and simplifying the routing algorithm. On the other hand, mesh is an asymmetrical topology in which the degree of a node depends on its location. Links closer to the center of the mesh experience higher traffic as compared to the links near the boundaries. However, the scalability of tori is poor as compared to meshes because of the presence of wrap around links, which usually have different weights as compared to the regular links.

In general, low dimensional meshes and tori topologies are preferred in designing a multicomputer because they have low node degrees and fixed-length channel wires, which make them more scalable than high-dimensional meshes and tori. They also have higher channel bandwidth per bisection density and lower blocking latencies, which result in lower communication latency [15].

In this thesis, multicast algorithms are designed for 2-dimensional meshes and 2-dimensional tori networks.

2.1.1.4 Passing of Messages

A message between the nodes of a network can be passed in three approaches: *unicast*, *broadcast* and *multicast*. These approaches can be explained in terms of a network as follows: Assume a network N consisting of n nodes or processors in which a message is passed from the source node s to a number of destination nodes d.

a). ***Unicast***: In this case, the message is passed from the source s to only one destination node at a time, which makes $d = 1$ and therefore the message is sent to each destination separately.

b). ***Broadcast***: In this approach, the message is sent to all other processors in the network irrespective of the requirement. Thus in this case $d = n - 1$, where n is the total number of nodes in the network.

c). ***Multicast***: In this case, message is sent to exactly d processors that actually need the message, thus $1 \leq d \leq n - 1$.

It is clear that unicasting and broadcasting are special cases of multicasting, which is a very basic operation in multicomputer. Memory updates in distributed shared memory system are also carried out by a series of multicast operations [9, 20].

2.1.1.5 Switching of Messages

Performance of a multicomputer system is highly dependent on the underlying communication network, which in turn is dependent on the switching of messages, also known as switching techniques. Switching is the mechanism by which a message is moved from the input channel and placed into the output channel. There are four widely known techniques for switching [16, 23].

a). ***Circuit Switching***: consists of two phases that are *circuit establishment phase* and *message transmission phase*. A path from the source to destination is reserved for the message that contains details about destination address and some control information. The circuit is released as soon as the tail of message reaches the destination.

b). ***Wormhole Switching***: for the purpose of transmission, the message is divided into a number of flits. The header governs the route and remaining flits follow the header in pipeline fashion without any delay on the specified route. As soon as a flit arrives at a node, the node forwards it to next node without waiting for other flits to arrive. If the output channel is busy and the header flit cannot pass through it, all other flits get stuck. This results in occupying buffer in the router on the constructed path and blocking other possible communications.

c). ***Store and forward***: message is divided into fixed-length packets and every channel has input and output buffers for an entire packet. Each packet is routed, independent of other packets, whenever destination node has enough space for one packet.

d). ***Virtual Cut-through***: packet is buffered at a transitional node if the required channel is busy, else it is immediately forwarded as in the case of *Store and forward* technique. All buffers along routing path are prevented from other communication requirements.

2.1.2 Multicast

2.1.2.1 Multicast Problems

Let the interconnection topology of a network with n nodes be denoted by a host graph $G(V, E)$, where each vertex V of graph G corresponds to a node of the network and each edge E of graph G corresponds to a communication link of the network. For a multicast communication model, let u_0 denote the source node and $u_1, u_2, \dots, u_k$ denote k destination nodes, where $1 \le k \le |V|$. The set $K = \{u_0, u_1, u_2, \dots, u_k\}$, which is a subset of $V(G)$, is called a multicast set. According to Lin and Ni [9], the multicast communication problems in a network can be formulated as four different graph theory problems: *Multicast Path (MP)* problem, *Multicast Cycle (MC)* problem, *Steiner Tree (ST)* problem, and *Multicast Tree (MT)* problem. These problems depend on the underlying communication model and routing technique. All these problems take a unique approach in reaching the solution of multicast in a network.

Multicast Path (MP) **Problem**

In some communication mechanisms, duplication of an incoming message in order to forward the message to multiple neighboring nodes involves high overhead and is usually unfavorable. Thus, the routing method does not allow each processor to duplicate the message passing by. In such case, the multicast routing problem is finding a shortest path starting from u_0 and visiting all k destination nodes. The optimization problem is to find an *optimal multicast path (OMP)* and is formally defined in [9] as:

Definition 3 (*Optimal Multicast Path*): A *multicast path (MP)* $(v_1, v_2, \dots, v_n)$ for multicast set K in G is a sub-graph $P(V, E)$ of G, where $V(P) = \{v_1, v_2, \dots, v_n\}$ and $E(P) = \{(v_i, v_{i+1}) : 1 \leq i \leq n - 1\}$, such that $v_1 = u_0$ and $K \subseteq V(P)$. An *OMP* is a *MP* with the shortest total length.

Multicast Cycle (MC) Problem

Reliable communication is essential in a message passing system. Usually, a separate acknowledgment message is sent from every destination node to the source node upon receipt of a message. One way to avoid the sending of K separate acknowledgment messages is to have the source node itself receiving a copy of the message it initiated after all destination nodes have been visited. Thus, the multicast routing problem is finding a shortest cycle, called *optimal multicast cycle (OMC)* for K. *OMC* is defined in [9] as:

Definition 4 (*Optimal Multicast Cycle*): A multicast cycle $(v_1, v_2, \dots, v_n, v_1)$ for K is a subgraph $C(V, E)$ of G, where $V(C) = (v_1, v_2, \dots, v_n)$ and $E(C) = \{(v_n, v_1), (v_i, v_{i+1}) : 1 \le i \le n-1\}$, such that $K \subseteq V(C)$. An *OMC* is an *MC* with shortest total length.

Steiner Tree (ST) Problem

Both *OMC* and *OMP* assume that any node will not replicate the message during transmission. However, message replication can be implemented by using some hardware approach**.** If the main objective is to minimize *traffic*, message replication should not be done and thus the multicast problem becomes the well-known *Steiner tree problem***.** Formally, the *ST* problem is defined in [9] as follows:

Definition 5 (*Minimal Steiner Tree*): A Steiner tree, $S(V, E)$, for a multicast set K is a subtree of *G*, such that $K \subseteq V(S)$. A *Minimal Steiner Tree (MST)* is a *ST* with a minimal total length.

Multicast Tree (MT) Problem

In the *ST* problem, it is not required to use the shortest path from the source to a destination. If the distance between two nodes is not a major factor to the *communication time*, such as circuit switching, the *ST* problem is appropriate. However, if the distance is the major factor to the *communication time*, such as the *store-and-forward* mechanism, then the objective is to minimize *time* first, and then *traffic*. The multicast communication problem is then modeled as an *optimal multicast tree (OMT)* problem and is defined in [9] as follows:

Definition 6 (*Optimal Multicast Tree*): An OMT, $T(V, E)$, for a multicast set K is a subtree of G, such that a) $K \subseteq V(T)$, b) $d_T(u_0, u_i) = d_G(u_0, u_i)$, for $1 \leq i \leq k$, and c) $|E(T)|$ is as small as possible.

Complexity of Multicast Problems

The complexity of each of the above optimization problems is directly dependent on the underlying host graph. Lin and Ni [9] proved a number of results about the complexity of multicast problems, which in this work are treated as theorems.

Theorem 1: The *OMC, OMP, MST* and *OMT* problems for *2*D mesh are all NP-complete.

Later Harutyunyan and Wang [4] concluded that the above theorem could be applied to higher dimensional meshes.

Theorem 2: The *OMC, OMP, MST* and *OMT* problems for *n*-dimensional ($n \geq 2$) mesh are all NP-complete.

2.1.2.2 Multicast Models

In reality multicasting can be implemented in different ways, which are referred as multicast models. In general, multicast models are classified into two categories: unicast-based and tree/path-based.

Unicast-based Multicast

Unicast-based multicast can be implemented in many ways. The simplest approach is to send an individual copy of message from the source node to every destination node. This technique is called separate addressing [19]. This approach is simplest but is costly on both time and traffic since it is completely sequential and makes no use of any previously used links.

Another approach is known as chain tree, in which the source node sends a copy of the message to only a subset of destinations. These destinations then forward the message to those destinations, which have not received the message yet. This process continues until all destinations receive the message, forming a logical message-passing tree composed of all destination nodes. Therefore this approach is efficient on the number of traffic links used.

Tree/Path-based multicast

In tree-based multicast, the hardware at each node supports multicast routing. Routing information with the set of destinations is embedded in the header of a packet. The source node sends the packet to a selected set of its neighboring nodes, each of which is able to extract the routing information without any intervention from the CPU and decides the next step. Depending on the routing information embedded in the header, the receiving nodes may simply accept the packet (a leaf destination) or may pass the message to one of its neighbors (a passerby node) or both (an intermediate destination node). In some cases a node (replicate node) may need to copy the message, modify the routing information in the header, and then pass it to several neighboring nodes. Consequently, the message is disseminated from node to node in a certain order until all destinations receive the message. All nodes and links involved in the distribution occur only once at a certain time, forming a tree in the host network rooted at the source node. In some cases, the message is passed along many paths starting at the source node as seen in [4].

Tree-based multicast is recognized for its high efficiency of both time and traffic [11, 20, 21, 25, 26]. Hardware support and high degree of parallelism in the message distribution significantly reduce the communication latency. In the multicast tree, destination nodes share as much common path as possible and none of the links occur twice. Therefore, the total traffic generated will be small. This approach outperforms the unicast-based multicast and has become a more promising multicast model for the new generation of multicomputers.

2.1.2.3 Multicast in Mesh and Torus Network

Due to the properties of mesh and torus networks as described in *Section 2.1.1.3*, they have become popular topologies in multicomputer systems. Compared to multicasting in an arbitrary network, multicasting in these networks is much simpler to design. As a result of geometric regularity of a mesh/torus network, multicast routing can be implemented at a fairly low cost. Nodes of a mesh/torus are arranged like an array, and connectivity and distance between them can be determined by comparing their coordinates. These characteristics significantly reduce the complexity of the algorithm, which otherwise will involve a lot of graph based searching. Moreover, length and bandwidth of each link is almost the same. Hence, same weight can be assigned to the link, which simplifies the calculation.

2.1.2.4 Multicast Evaluation Criteria

The main process of multicasting is to select a route to deliver a message from the source node to all the destinations in an efficient and economical way. Lin and Ni [9] proposed that *traffic* and *time* are two major parameters to evaluate multicast communication. *Traffic* refers to the number of links used to deliver the source message to all its destinations. *Time* is the message transmission time starting when the source sends out the message till the last destination receives it.

Good implementation of a multicast routing algorithm involves few contradicting requirements. One of the requirements states that the message should be sent from the source node to all destinations as quickly as possible, which minimizes the multicast *time*. Another requirement states that the multicast process should create as little *traffic* as possible. Substantial *traffic* not only costs resources but also causes contention, deadlock, and can increase the multicast *time*. Another important prerequisite is the complexity of the routing algorithm, which should be small. Large computation complexity will increase delay at each node and deteriorate the overall performance. Thus both parameters *time* and *traffic* are not totally independent, and obtaining optimal result for one parameter may worsen the result for the other.

From *Theorem 1*, *OMP*, *OMC* and *MST* problems, which try to obtain optimal multicast traffic, are *NP-complete*. Therefore, the problem of achieving optimal results for both time and traffic is *NP-hard*.

2.1.2.4.1 Evaluation of Multicast Traffic

Multicast traffic is calculated in the same way regardless of the underlying network topology, switching technology, and routing techniques. The multicast traffic is the total number of edges in the tree. Let $MT(V, E)$ define the multicast tree for routing a message from source u_0 to k destinations where V is a set of nodes in the tree and E is the set of edges in the tree,

Multicast Traffic = $|\mathrm{E}(\mathrm{MT})|$

Apart from *multicast traffic*, another parameter *additional traffic* is also widely used to evaluate traffic. It is essentially the difference between total traffic and the number of destinations [9]. Sending a message to k destinations needs at least k links, each of which delivers the message to one destination. So, the amount of traffic on top of the minimum necessary traffic is considered additional traffic. It reflects the efficiency of traffic better than total traffic, thus

Additional Traffic = $|\mathrm{E}(\mathrm{MT})| - k$

2.1.2.4.2 Evaluation of Multicast Time

Multicast time is the message transmission time of a multicast starting when the source node sends out the message till the last destination receives the message. Communication latency depends on the underlying network technology especially the switching techniques. Criteria of evaluation of multicast time vary from one switching technology to another [4]. Only *store-and-forward* switching technology is discussed in detail, which is one of the most commonly used switching technologies in multicomputers, and is used to design new algorithms later in this thesis.

For Unicast-based Multicast

Unicast-based multicast is accomplished through a series of unicasts between destinations in the order designated by a logical multicast tree. Each step of unicast takes approximately the same amount of time. Therefore, the unicast-based multicast latency is actually decided by the maximum number of unicast steps needed to send the message to all destinations. Assume the number of unicasts required to deliver the message from source node u_0 to destination nodes $v_1, v_2, \dots, v_k$ is $n_1, n_2, \dots, n_k$ respectively, then

Multicast Time = $\max\{n_1, n_2, \dots, n_k\}$

For Tree/Path-based Multicast

Evaluation of multicast latency for tree-based multicast is more complicated. Assume a message has to be delivered to k destinations and the length of the path from source node to the destination nodes $d_1, d_2, \dots, d_{k-1}, d_k$ in the multicast tree are $l_1, l_2, \dots, l_{k-1}, l_k$ respectively.

The network latency in store-and-forward switched networks is proportional to the distance, and can be counted in hops, which represent the time for a message to be transmitted from a node to its neighboring node. In a network with all-port architecture where the message can be transmitted from a node to all its neighbors simultaneously, the multicast time will simply be the length of the longest path to a destination, i.e.

$$\textbf{\textit{Multicast time}} = \max\{l_1, l_2, \dots, l_{k-1}, l_k\}$$

But in networks with one-port architecture, if a branch node has to send a message to two of the neighboring nodes, *hops* waited at the branch nodes have to be counted. Assume on the path to destination node d_i, there are n branching nodes and the *hops* waited at each branching node is $h_1, h_2, \dots, h_{n-1}, h_n$, then the total *hops* waited at branching nodes for d_i is w_i where

$$w_i = \sum_{j=1}^{n} h_j$$

Multicast time will be the maximum number of *hops* taken by a message to reach all the destinations.

$$\textbf{\textit{Multicast Time}} = \max\{l_1 + w_1, l_2 + w_2, \dots, l_{k-1} + w_{k-1}, l_k + w_k\}$$

2.2 Review of Previous Studies

Several algorithms have been developed and papers have been published related to multicast in mesh-connected multicomputers since early 1990. Following is a brief review of the history of the studies in this field.

Basic Theories and Early Studies

In 1993, Lin and Ni [9] introduced multicast communication in multicomputer systems in detail and gave the formal definitions of the multicast problems. They have proved a series of result such as NP-completeness of multicast problems, defined the criteria for evaluation of performance of multicast routing, and proposed several heuristic multicast algorithms.

Most of the first generation multicomputers used store-and-forward switching and adopted the hypercube topology because of very short distance among any pair of nodes. But later on, wormhole routing emerged as a more popular switching technology and low dimensional meshes gradually replaced hypercube.

Later, Ni and McKinley [16] conducted a comprehensive survey on wormhole routing. As wormhole routing is particularly susceptible to deadlock, many approaches to ensure deadlock-free routing are proposed. In 1998, Mohapatra [15] discussed different techniques for improving the performance and reliability of wormhole routing.

Approaches on Unicast-based Multicast

Mckinley *et al* [13] proposed efficient algorithms to implement multicast communication in wormhole-routed meshes, which delivers a multicast message to $m-1$ destinations in $\lceil log_2 m \rceil$ message passing steps. In 1995, Robinson *et al* [20] presented a related version of the algorithm for torus networks, which also achieved the $\lceil log_2 m \rceil$ optimal result.

However, unicast-based multicasts were not proficient due to low parallelism, lack of traffic optimization, and considerable software involvement.

Approaches on Path-based Multicast

Lin and Ni [10] proved that path-based multicast routing outperforms tree-based multicast routing in wormhole networks. Further they proposed dual-path and multi-path algorithms for 2D-mesh and hypercube topology. Both algorithms use a Hamiltonian path to guide the routing of message with as short paths as possible. Because routing has to observe the order of Hamiltonian path, the algorithm generates excessively huge traffic. In 2011, Kuperman *et al* [8] proposed algorithms aimed at path-based multicast routing for mesh networks. These algorithms were mainly focused on partial protection from network failure when few links or nodes fail.

Approaches on Tree-based Multicast

In 1997, Yang and King [31] proposed a multicast routing scheme for messages of arbitrary length in wormhole routed 2-D meshes. This scheme constructed a *quad-branch multicast* (QBM) tree for transmitting the given multicast. To maintain QBM, routers were designed and initialized in a particular way.

Harutyunyan and Liu [3] in 2003 suggested two tree-based multicasting algorithms for mesh and torus networks: *VH* (Vertical-Horizontal) and *DIST* (Distance). They came up with following results, *DIST* algorithm generates less traffic than *VH* algorithm, but at a price of much larger multicast time and computation complexity. Whereas *VH* obtained very low computational complexity and optimal multicast time but creates a very large amount of traffic.

Later Harutyunyan and Wang [4] improved the previous time optimal *VH* algorithm and proposed *DIAG* (Diagonal). The main strategy was to make a shortest path multicast algorithm and to eliminate the bad scenarios in *VH* that generate excessive traffic. They suggested another tree-based shortest path multicast routing algorithm known as *DDS* (Dimensional Distance Sorted). It was designed to obtain near optimal multicast time and to further reduce the large multicast traffic of the previous algorithms. *DIAG* and *DDS* generate much less traffic than *VH*, but can obtain nearly the same multicast time and complexity. *VH*, *DIST* and *DIAG* will be discussed in detail in coming sections.

In 2010, [21] presented the first network-on-chip based on a mesh topology, which supports adaptive and deadlock-free tree-based multicast routing. In 2012, [25] proposed a dual-tree-based multicast routing protocol for mobile *ad hoc* networks. This approach generates more traffic and achieves multicast in less time.

Due to the properties of mesh and torus topologies, multicast routing that obtains *optimal time* is achievable at a low complexity.

2.2.1 VH and DIAG Algorithm

In *VH* algorithm, the message is routed in strictly dimensional order to each destination node. Therefore, the message is first transmitted from the source node along the lowest dimension, and then turns to the next lowest dimension. The process continues until the message is received by all the destinations [3]. *VH* is a time optimal algorithm and gives little attention to the multicast traffic generated.

Example 1: In a 2-D mesh of size (8×8), construct a multicasting tree for a multicast that sends a message from source node $s(0,0)$ to a set of destination nodes $\{(4,6),(6,6),(0,2),(4,0),(3,0),(7,4)\}$.

Figure 3 shows a 2-D mesh of size (8×8), with source node $s(0,0)$ and destination set $K = \{(4,6),(6,6),(0,2),(4,0),(3,0),(7,4)\}$ as given in *Example 1*.

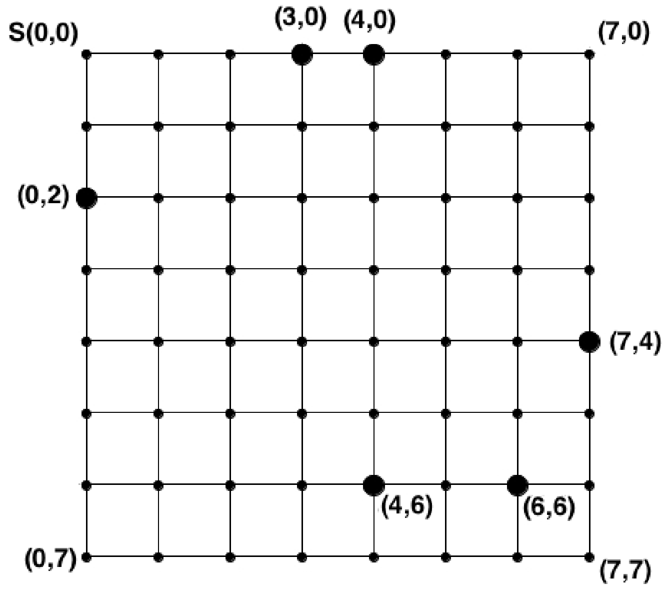

Figure 3 Mesh of *Example 1*

If *VH* algorithm is used to construct a multicast tree for *Example 1*, it takes 25 traffic links and 12 time units to distribute the message to all the destinations in K. *Figure 4* shows the multicast tree constructed through *VH* algorithm for this example.

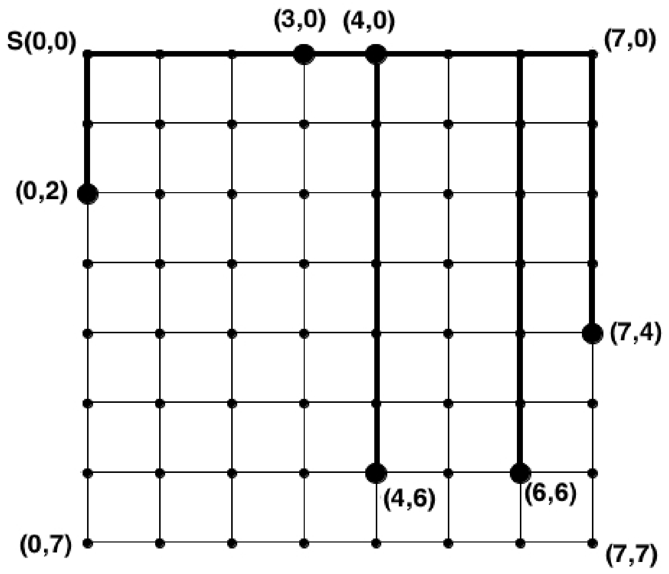

Figure 4 Example of VH Algorithm

The *DIAG* algorithm suggests that the major routing path needs to travel along both dimensions concurrently, instead of just one direction. Hence major routing path in *DIAG* advances along both dimensions alternatively in small steps until both dimensions reach their maximum coordinates [4].

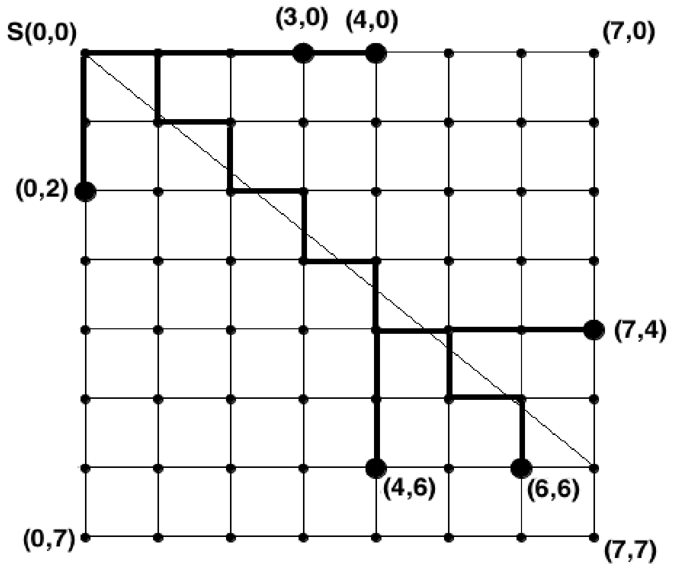

Figure 5 Example of DIST Algorithm

Figure 5 shows the multicast tree constructed by using the *DIAG* algorithm for the same example. It uses 21 traffic links and 12 time units to distribute the message to all the destinations in K.

2.2.2 DIST Algorithm

In *DIST* algorithm, distance for each destination i is calculated by adding all of its coordinates and assigned to D_i where i is one of the destination nodes. After this, source node is connected to the destination with minimal distance. Then destination with next minimal distance is chosen, and it gets connected to the existing multicast tree with a shortest path. This process is repeated until every destination node is included in the multicast tree.

If *DIST* algorithm is used to construct a multicast tree for *Example 1*, it takes 17 traffic links and 17 time units to distribute the message to all the destinations in K. *Figure 6* shows the multicast tree constructed through *DIST* algorithm.

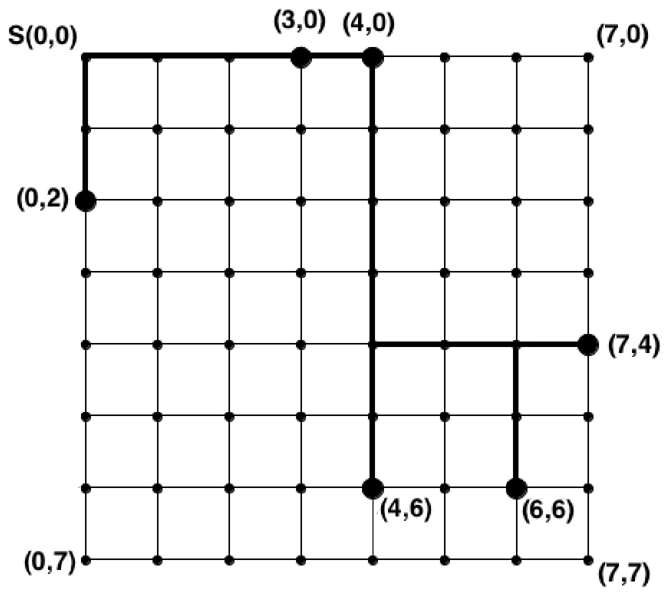

Figure 6 Example of DIST Algorithm

Chapter 3

Proposed Algorithms

3.1 Problem and Multicast Communication Model

When a multicomputer of mesh or torus network is visualized as a graph, the multicast problem in them reduces to one of the graph theory problems as discussed in *Section 2.1.2*.

3.1.1 The Problem

Time and *traffic* being indirectly related, makes the problem of achieving results that are optimal for both time and traffic, an *NP-hard problem*. A good approach would be to design an algorithm that obtains optimal result on one parameter and then reduce the value of other parameter as much as possible with a reasonable complexity.

If the algorithm tries to obtain minimal traffic first, the problem would become *Minimal Steiner Tree (MST)* or *Optimal Multicast Path (OMP)*. However if the algorithm minimizes the time first and then tries to reduce the traffic, the problem would become *Optimal Multicast Tree (OMT)*.

This study uses *OMT* to design efficient multicasting algorithms. Hence the problem would be to find an *Optimal Multicast Tree (OMT)* in mesh and torus networks.

An *OMT* delivers message along a common path as much as possible, and then branches the message to each destination node that results in a tree-like structure. Therefore, an *OMT* can be easily converted to shortest path problem in a mesh network.

The formal definition of *OMT* in mesh or torus network can be given as:

Definition 7 (*Optimal Multicast Tree (OMT) in Mesh or Torus Networks*): Given, a mesh or torus network M, and a source node u_0 to k destination nodes set $K = \{u_1, \ldots, u_k\}$, find the OMT, $T(V,E)$, such that a) T is a subgraph of M, b) $K \subseteq V(T)$, c) $d_T(u_0, u_i) = d_M(u_0, u_i)$, for $1 \leq i \leq k$, and d) $|E(T)|$ is as small as possible.

Solution to Multicast in 2-D Mesh

The *2*-dimensional mesh fits into the positive quadrant of the *2*-dimensional coordinate space with their origins overlapping. Each node of the mesh is located at a point that is identified by the location corresponding to a *2*-coordinate vector, and each pair of neighboring nodes are connected with a line that represents a link between them.

The distance between two nodes can be calculated as the addition of the absolute differences of their coordinates in their respective axes. For example distance between a node N_1, located at coordinate $(2,3)$, and N_2, located at $(4,7)$, will be $|2-4| + |3-7| = 6$ units.

In reality any node can be the source node u_0 that has to pass the message to a set of destination nodes. In order to simplify the mathematical model and simulations, it is assumed that the source node is located at the origin of mesh. This assumption has been proved legitimate in [4].

Solution to Multicast in 2-D Torus

Multicast problems in torus are similar to those in a mesh. In a 2D mesh, delivery of message always follows the positive direction, which depends on the position of the nodes, whereas in a 2D torus it can follow either positive or negative direction.

A 2D torus of size $(m \times n)$ can be divided into four zones as shown in *Figure 7* and each node (x, y) where $0 \leq x \leq m-1$ and $0 \leq y \leq n-1$, belongs to one of them.

$$zone_1: \{ (0,0) \Leftrightarrow (\lceil m/2 \rceil - 1,\ \lceil n/2 \rceil - 1) \}$$

$$zone_2: \{ (m-1,0) \Leftrightarrow (\lceil m/2 \rceil,\ \lceil n/2 \rceil - 1) \}$$

$$zone_3: \{ (0,n-1) \Leftrightarrow (\lceil m/2 \rceil - 1, \lceil n/2 \rceil) \}$$

$$zone_4: \{ (m-1,n-1) \Leftrightarrow (\lceil m/2 \rceil,\ \lceil n/2 \rceil) \}$$

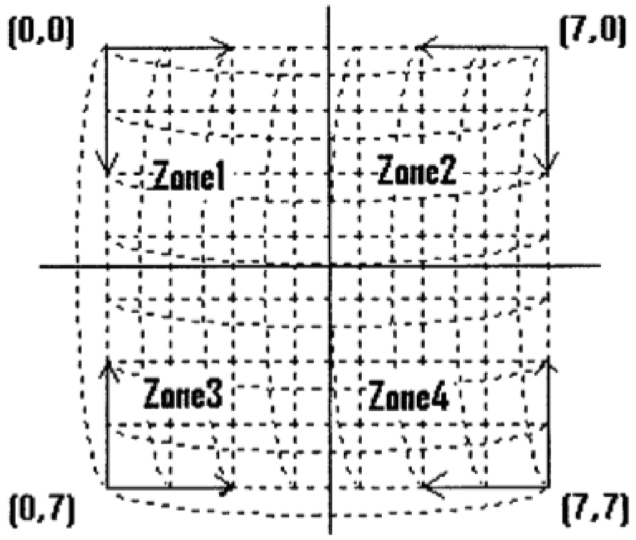

Figure 7 Division of 2D Torus Network

A torus is partitioned into zones to use the wraparound links present, which helps in disseminating the message faster as compared to a mesh network. The division of 2D torus reduces the problem of multicast in 2D torus to a problem of multicast in four sub-meshes. At *time* unit one, message is sent from the source node $s(0,0)$ to $(m-1,0)$ and at next time unit message is sent from $s(0,0)$ to $(0,n-1)$ as well as from $(m-1,0)$ to $(m-1,n-1)$.

In $zone_1$, message is routed from node $s(0,0)$ to all the destinations in it. Similarly in $zone_2$, $zone_3$ and $zone_4$ message routing is done from $(m-1,0)$, $(0,n-1)$ and $(m-1,n-1)$ respectively. Thus the problem of multicast in torus is reduced to the problem of multicast in four different sub-meshes.

3.1.2 Multicast Communication Model

The multicast communication model for any mesh network (*N*), that has a source node or originator u_0 and the message should be passed to k destination nodes of set $K = \{u_1, \dots, u_k\}$ should satisfy the following conditions:

- During each time unit one processor may transmit the message to only one of its neighboring nodes.
- During each time unit a processor may receive at most one message.
- During each time unit a message may be transmitted over different links simultaneously.
- The communication process ends when all the destinations in set K receive the anticipated message.

3.2 PAIR Multicast Algorithm

In-depth analysis of *VH* and *DIAG* algorithms shows that both of these algorithms have limitations in some scenarios. *VH* does not perform well if many destinations are located at the bottom of the mesh and *DIAG* does not perform well if destinations are located at a far distance from the diagonal of the mesh.

PAIR is a time-optimal algorithm and is designed to further reduce the traffic of the previous tree-based time-optimal *DIAG* algorithm. The algorithm first finds a pair of nodes from the destination set, in a way that the pair has minimum distance from the source node, and then constructs a multicast tree to include these two nodes. The next two nodes are then chosen in the same manner and the process is repeated until every destination node is included in the multicast tree.

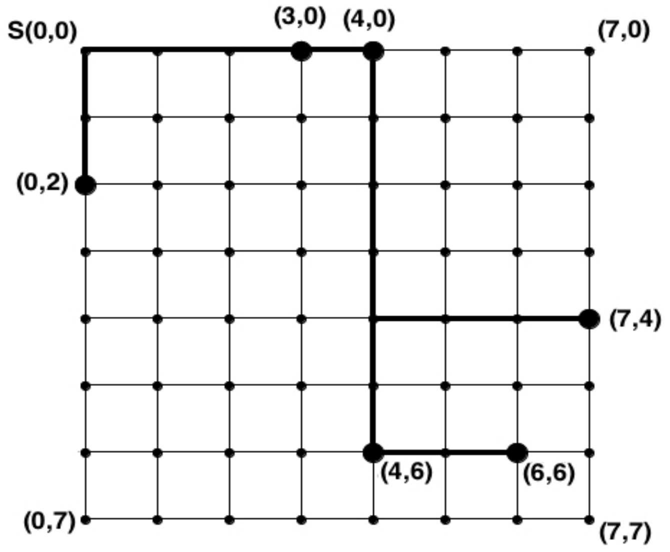

Figure 8 Example of PAIR Algorithm

Figure 8 shows the multicast tree constructed using the *PAIR* algorithm on *Example 1* as described in *Section 2.2.1*. The *traffic* generated to achieve multicast in this example is 17 links and *time* taken by the last destination to receive the message is 12 hops. *VH* used 25 traffic links and *DIAG* used 21 traffic links for the same example as shown in *Figure 4* and *5*. Therefore, it is observed that the *PAIR* algorithm matches the optimal multicast time of *VH* and *DIAG* and significantly reduces the traffic.

3.2.1 Concept behind PAIR Algorithm

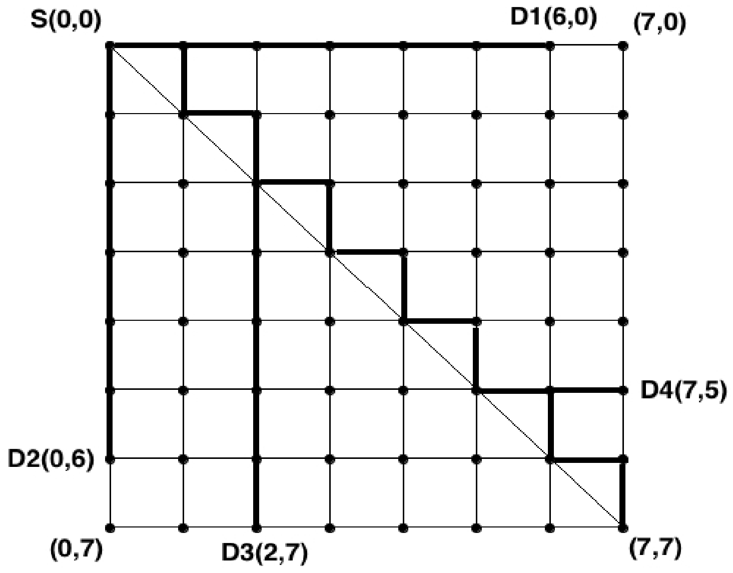

Figure 9 Bad Scenario of DIAG

Figure 9 shows an example of poor performance of *DIAG* algorithm in a mesh network where the destination nodes D_1, D_2 and D_3 are located far from the diagonal. In this scenario, *DIAG* generates 29 traffic links to achieve multicast.

PAIR focuses on reaching the destination nodes using a shortest path from the source node rather than generating a diagonal structured multicast tree. Hence, PAIR reduces the multicast traffic, while keeping the optimal multicast time. A pair of nodes is considered at a time, thus the name *PAIR* is been assigned to this algorithm.

The *PAIR* algorithm finds a pair of nodes such that one of the nodes has minimum x value (x_{min}, y) and another has minimum y value (x, y_{min}), where (x_{min}, y) and (x, y_{min}) are the coordinates of two nodes from the destination set of a 2D mesh network. A multicast tree is constructed from the source to node (x_{min}, y_{min}) and this node is called an *intermediate node*. The destinations (x_{min}, y) and (x, y_{min}) are then connected to the intermediate node. This process is repeated until every destination node is included in the existing tree.

Approach behind PAIR Algorithm

Figure 10 shows a mesh network and some destination nodes. A message from source node $s(0, 0)$ should be sent to all the destination nodes.

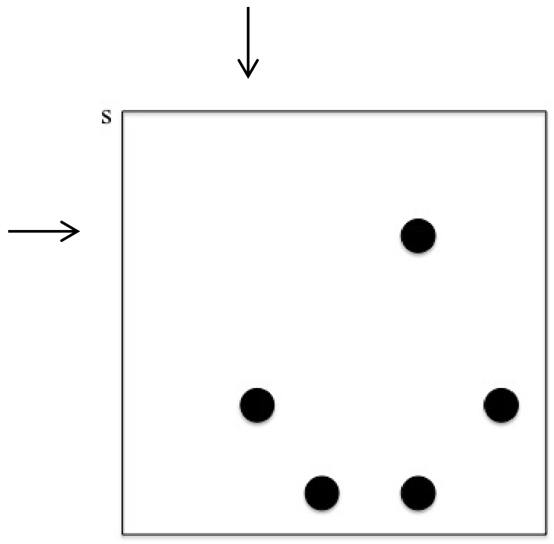

Figure 10 Approach behind PAIR Algorithm

The mesh is scanned in both dimensions to locate nodes that have x_{min} and y_{min} in order to find the intermediate node (x_{min}, y_{min}). Once the intermediate node is found, it can be assured that there are no more destinations located between $(0,0)$ and $(x_{min}-1,0)$ as well as between $(0,0)$ and $(0,y_{min}-1)$.

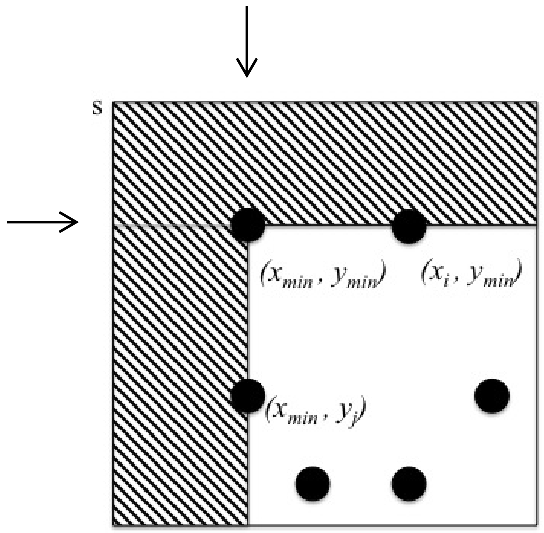

Figure 11 Reducing the Problem Size

When node (x_{min}, y_{min}) receives the message, the problem size is reduced to the non-shaded area, with new source node located at last intermediate node, which is (x_{min}, y_{min}) in this case.

3.2.2 PAIR Algorithm in 2D Mesh

This section presents formal description of *PAIR* multicast algorithm in *2*D meshes.

Algorithm 1 ***(PAIR Algorithm in 2D Mesh):*** In a *2*D mesh network of size $(m \times n)$, given a source node $s(0,0)$ and a set of k destination nodes $\{(x_1,y_1),(x_2,y_2) \ldots (x_i,y_i)\}$, where $0 \le x_i \le m-1$, $0 \le y_i \le n-1$ and $i=[1,k]$. Multicasting requires message to be sent from the source node s to all k destinations.

The main steps of *PAIR* multicast routing algorithm for a 2D mesh are

Step 1. Find a pair of destination nodes A and B, where node A has the minimum x value x_{min} in the destination set and node B has the minimum y value y_{min} in the destination set.

Step 2. Construct a multicasting tree between the source node and the node (x_{min}, y_{min}).

Step 3. Connect node A and B to node (x_{min}, y_{min}).

Step 4. Repeat *steps 1-3* until every destination is included in the multicasting tree.

To simplify the *Step 1* of the algorithm, destination nodes are *arranged*, which eases the pairing of nodes.

Definition 8 (Arrange destination nodes): Given a set of k destination nodes in any order, destination nodes are arranged as $\{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$ where $x_1 \le x_3 \le x_5 \le \cdots \le x_{k-1}$ and $y_2 \le y_4 \le y_6 \dots \le y_k$.

It is possible that, x_{min} and y_{min} may belong to a similar node. In such cases the node is paired up with itself as $\{(x_i, y_i), (x_i, y_i)\}$ and thus the intermediate node will be (x_i, y_i) that is the node itself.

A multicast tree is constructed between source and the intermediate node using a shortest path and then destination nodes are connected to the intermediate node using the shortest path. This process is repeated until all the destinations receive the message.

Example 2: In a 2D mesh network of size (10×9), construct a multicast tree for a multicast that sends a message from the source node $s(0,0)$ to a set of destination nodes $\{(3,2),(9,3),(8,5),(4,7),(7,7),(2,8),(5,8),(9,8)\}$.

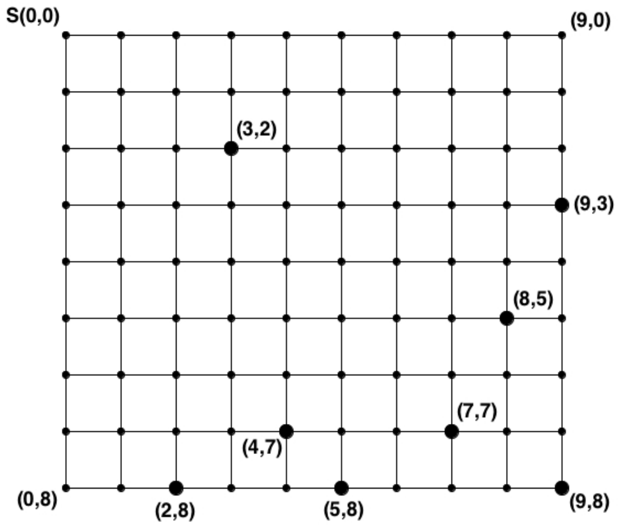

Figure 12 Mesh of *Example 2*

Figure 12 shows 10×9 mesh network with source node $s(0,0)$ and all destinations. Destination nodes are *arranged* as defined in *Definition 8*. The arranged destination set is $\{(2,8),(3,2),(4,7),(9,3),(5,8),(8,5),(7,7),(9,8)\}$.

After destination nodes are *arranged*, pairing of destination nodes is done in order to calculate intermediate nodes

$\{(2,8),(3,2)\} \Rightarrow (2,2)$

$\{(4,7),(9,3)\} \Rightarrow (4,3)$

$\{(5,8),(8,5)\} \Rightarrow (5,5)$

$\{(7,7),(7,7)\} \Rightarrow (7,7)$

$\{(9,8),(9,8)\} \Rightarrow (9,8)$

The source node $s(0,0)$ is connected to first intermediate node, which is $(2,2)$ marked by **X** using a shortest path as shown in *Figure 13*.

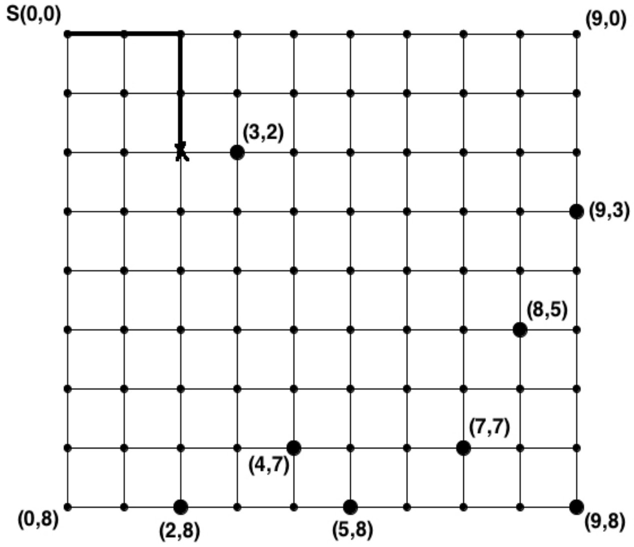

Figure 13 Intermediate Node receives the message

When intermediate node $(2,2)$ receives the message, connect destination nodes $(3,2)$ and $(2,8)$ to it. This process is repeated until all the nodes in the destination set receive the message as shown in *Figure 14*.

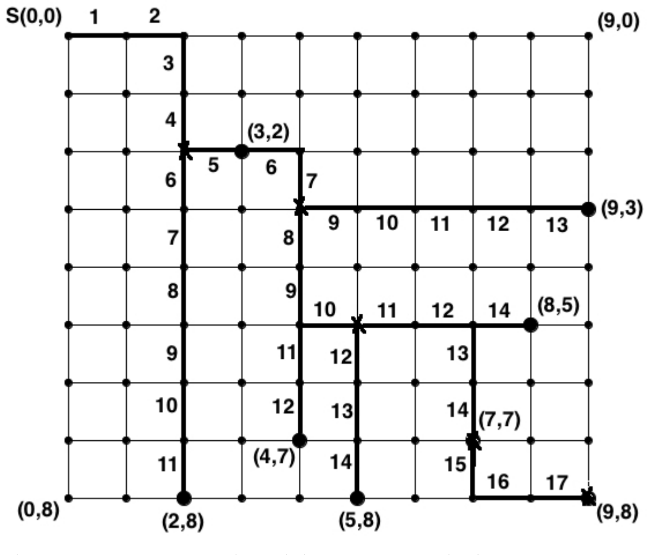

Figure 14 PAIR Algorithm on Mesh for *Example 2*

Figure 14 shows complete multicasting scheme on a 10×9 mesh network with a given set of destinations. The number of links used to multicast a message from the source to all destinations is 34 and time taken to achieve this multicast is 17 hops.

Proposition 1**:** Time complexity of $PAIR$ algorithm in *2*D mesh is $O(kD)$, where k is the number of destination nodes and D is the diameter of the mesh network.

Proof: Destinations nodes are *arranged* in $O(k\ logk)$ and pairing is done in linear time. Time complexity for each iteration in selecting two destination nodes is $O(k)$ and finding the path from existing multicast tree to the intermediate node is $O(kD)$.

Thus, the overall time complexity of the $PAIR$ algorithm becomes $O(k\ logk + kD) = O(kD)$ as $D > logk$.

Proposition 2**:** In the multicast tree constructed by *PAIR* algorithm, the path from the source node to any destination node is a shortest one.

Proof: Intermediate node is connected to the source node using a shortest path, and destinations use shortest path to reach the intermediate node and so on. This ultimately results in shortest path from source node to any destination node.

Proposition 3**:** *PAIR* multicast routing algorithm is deadlock free.

Proof: There is no deadlock within a single multicast since the message is routed along a multicast tree. For several simultaneous multicasts, *PAIR* algorithm uses dimensional-ordered XY routing to transmit message between any two nodes, which will assure no deadlock cycle [16].

Proposition 4**:** The bound of *PAIR* algorithm's multicast traffic in *2*D mesh network is: $D_{max} \leq Traffic_{PAIR} \leq\ n^2 - 1$, where D_{max} is the farthest destination node and n is the size of the mesh.

Proof: Assume a mesh network of size $(n \times n)$ with k destination nodes is given. Multicasting has to be done from source node $s(0,0)$ to all k destination nodes.

If all the k destination nodes are *arranged* as discussed in *Definition 8*, then as per the definition, the following conditions are true

$x_1 \leq x_3 \leq x_5 \ldots \leq x_{k-1}$ and $x_{2i-1} \leq x_{2i}$, for $i = [1, \frac{k}{2}]$

and

$y_2 \leq y_4 \leq y_6 \ldots \leq y_k$ and $y_{2i-1} \geq y_{2i}$, for $i = [1, \frac{k}{2}]$

Figure 15 has 6 *arranged* destination nodes,

First, destination nodes with coordinates (x_1, y_1) and (x_2, y_2) are used to calculate the coordinates of the intermediate node as (x_1, y_2).

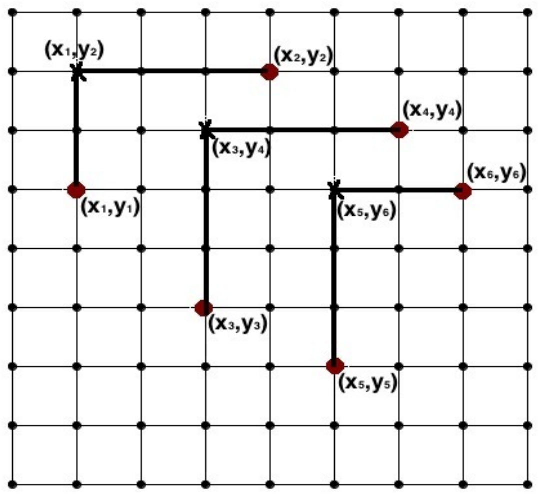

Figure 15 Traffic of PAIR Algorithm

To achieve multicast for two destination nodes, the total traffic generated would be

$x_1 + y_2 + |y_2 - y_1| + |x_2 - x_1|$

$\Rightarrow x_1 + y_2 + (y_1 - y_2) + (x_2 - x_1)$ as $y_1 \geq y_2$ and $x_2 \geq x_1$

$\Rightarrow x_2 + y_1$ ……………………………………………………………………………….(1)

For next 2 nodes with coordinates (x_3, y_3) and (x_4, y_4), the total traffic generated for a total of 4 nodes would be

$(1) + |x_3 - x_1| + |y_4 - y_2| + |y_4 - y_3| + |x_4 - x_3|$

$\Rightarrow x_2 + y_1 + (x_3 - x_1) + (y_4 - y_2) + (y_3 - y_4) + (x_4 - x_3)$ as $x_3 \geq x_1$, $y_4 \geq y_2$, $y_3 \geq y_4$ and $x_2 \geq x_1$

$\Rightarrow x_2 + y_1 - x_1 - y_2 + y_3 + x_4$

$\Rightarrow x_4 + y_3 + (x_2 - x_1) + (y_1 - y_2)$………………………………………..…………..(2)

Proceeding in the same manner, the total traffic generated for a total of 6 nodes would be

$(2) + |x_5 - x_3| + |y_6 - y_4| + |y_6 - y_5| + |x_6 - x_5|$

$\Rightarrow x_6 + y_5 + (x_4 - x_3) + (x_2 - x_1) + (y_3 - y_4) + (y_1 - y_2)$………..………………(3)

In a generalized form, the total traffic generated for k *arranged* destination nodes would be

$$Traffic_{PAIR} = x_k + y_{k-1} + \sum_{i=1}^{\frac{k}{2}-1}((x_{2i} - x_{2i-1}) + (y_{2i-1} - y_{2i}))$$

In the best case when all other destination nodes are located on a shortest path from the source node to the destination node with distance D_{max} as shown in *Figure 16*, the traffic generated would be D_{max}.

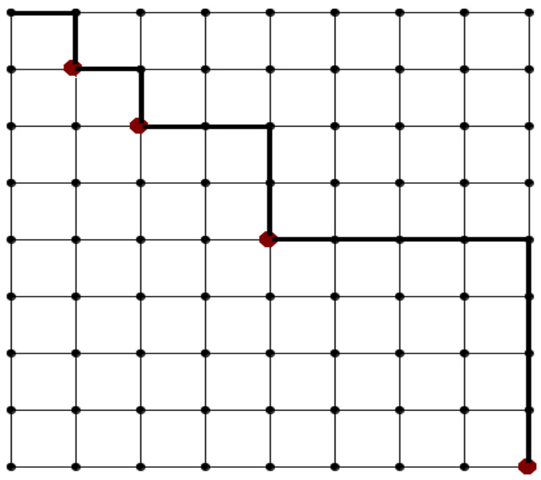
Figure 16 Best Case of PAIR Algorithm

In the worst case as shown in *Figure 17*, when all the destination nodes are located at the end of the mesh, for example nodes with coordinates $(0, n-1)$, $(n-1, 0)$, the number of links generated for each of the destinations would be $n-1$. Similarly for destination nodes with coordinates $(0, n-2)$ and $(n-2, 0)$, the number of links generated would be $n-2$ for each node.

Also to connect all the intermediate nodes, $n-1$ links would be generated.

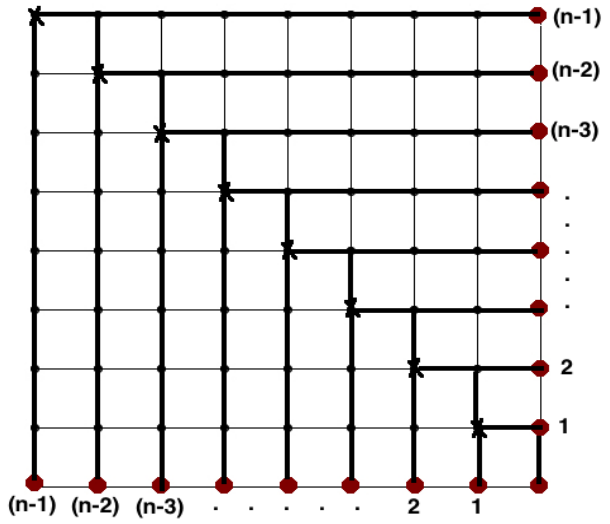

Figure 17 Worst Case of PAIR Algorithm

Thus, total traffic generated in worst case would be

$$Traffic_{PAIR} = 2(1 + 2 + \dots\dots\dots\dots\dots\dots\dots\dots\dots + (n-1)) + (n-1)$$

$$Traffic_{PAIR} = n^2 - 1$$

Hence the traffic bound of *PAIR* algorithm becomes

$$D_{max} \leq Traffic_{PAIR} \leq n^2 - 1$$

Proposition 5**:** The bound of *PAIR* algorithm's multicast time in 2D mesh is $D_{max} \leq Time_{\ Pair} \leq 2D_{max} - 1$, where D_{max} is the farthest destination node.

Proof: If a node needs to send the message to several neighbors, it does so sequentially which results in delay hops. The best multicast time is obtained when the path from the source node to the farthest destination takes the highest transmission priority direction at a branching node and is equal to D_{max}. In other case, when there is more than one destination located at a distance of D_{max}, there will be an increase in the multicasting time because the branching node can send the message to only one destination at a time. Therefore, multicast time depends on the number of destinations that have D_{max} distance. Maximum number of destinations that can have equal distance from the source node will be located on a diagonal of the mesh that does not include the source node itself. Thus the number of nodes that can have D_{max} distance will always be less or equal to D_{max}. Hence the upper bound becomes $2D_{max} - 1$.

3.2.3 PAIR Algorithm in 2D Torus

This section discusses how *PAIR* algorithm can be applied to *2*D tori.

Algorithm 2 ***(PAIR Algorithm in 2D Torus):*** In a *2*D torus network of size $(m \times n)$, given a source node $s(0,0)$ and a set of k destination nodes $\{(x_1, y_1), (x_2, y_2) \dots (x_i, y_i)\}$, where $0 \le x_i \le m-1$, $0 \le y_i \le n-1$ and $i = [1, k]$. Multicasting requires message to be sent from the source node s to all k destinations.

The main steps of *PAIR* multicast routing algorithm for a *2*D torus are:

Step 1. Divide the torus T into four sub-meshes M_1, M_2, M_3, M_4 as discussed earlier.

Step 2. Partition the destination set D into four subsets D_1, D_2, D_3, D_4 which contain the destination nodes in sub-mesh M_1, M_2, M_3, M_4 respectively.

Step 3. Assume $s_1(0,0)$ as the source node and origin of sub-mesh M_1, D_1 as the set of destination nodes, apply *PAIR* algorithm in sub-mesh M_1 and get the multicast sub-tree MT_1. Similarly apply the *PAIR* algorithm for M_2, M_3, M_4 to get multicast sub-tree MT_2, MT_3, MT_4.

Step 4. Assemble the multicast sub-trees MT_1, MT_2, MT_3, MT_4, into the overall multicast tree MT that is rooted at the source node.

Example 3: In a 2D torus network of size (8×8), construct a multicast tree using *PAIR* algorithm for a multicast that sends a message from the source node $s(0,0)$ to a set of destination nodes $\{(4,7),(7,5),(0,3),(2,3),(5,1),(6,3),(1,0),(3,0)\}$.

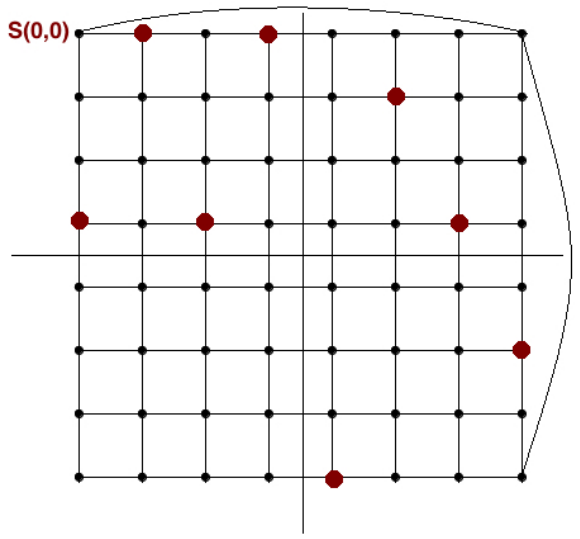

Figure 18 Torus of *Example 3*

Figure 18 shows how a 2D torus is divided into 4 different sub-meshes. Now PAIR algorithm for 2D mesh is applied separately on required sub-meshes, a multicast tree shown in *Figure 19* is generated.

Multicast traffic is 21 links and multicast time is 6 hops.

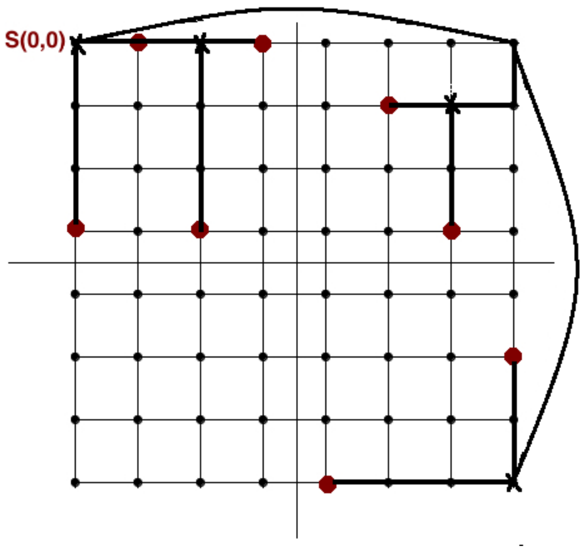

Figure 19 PAIR Algorithm on Torus of *Example 3*

3.3 MIN Multicast Algorithm

For any efficient multicasting algorithm minimum multicast time is desired, however this does not always guarantee the best performance. For certain problems, minimal routing algorithms are required that can utilize the available network bandwidth efficiently and cause less communication congestion.

MIN is a tree-based multicast routing algorithm for store-and-forward switched mesh networks. It is designed to further reduce the *multicast traffic* of time optimal *PAIR* algorithm and to obtain near optimal *time*. The *MIN* algorithm takes the advantage of the *PAIR* and *DIST* algorithms. *DIST* is an efficient algorithm to reduce the multicast traffic.

3.3.1 Concept behind MIN Algorithm

There are some scenarios where *PAIR* algorithm generates excessive traffic.

Figure 20 explains one of such scenario, in a 2D mesh of size (10×9), a multicast tree is constructed from the source node $s(0,0)$ to a set of destination nodes $\{(9,3),(3,1),(2,8),(4,8)\}$. According to *PAIR* algorithm $(4,8)$ should get connected to the multicast tree via $(4,3)$.

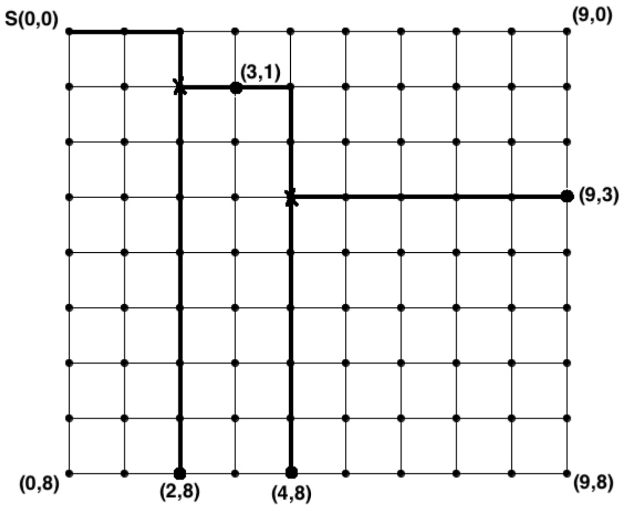

Figure 20 Bad Scenario of PAIR

However this is not efficient on traffic as $(4,8)$ should be connected to multicast tree via $(2,8)$. Number of traffic links used in this achieving multicast using *PAIR* is 24.

In order to reduce this traffic, *MIN* focuses on reaching the destination nodes using the shortest path from the existing multicast tree. In *MIN*, the logic of generating many intermediate nodes is discarded, and only 1 intermediate node is generated, and thereafter rest of the *arranged* destinations are connected to the existing multicast tree using shortest path approach, as shown in *Figure 21*.

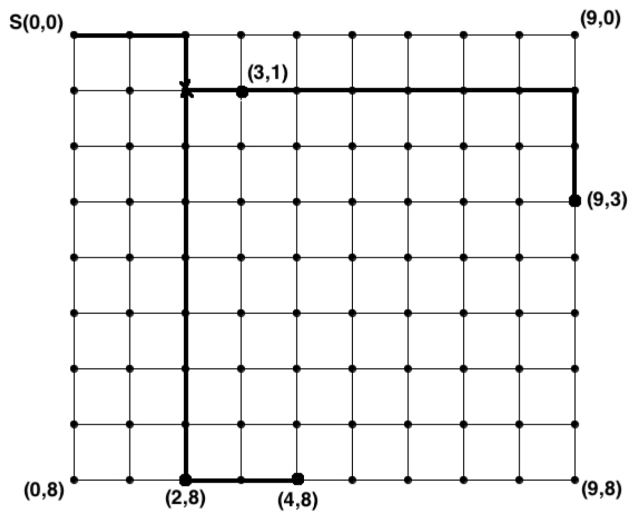

Figure 21 Example of MIN Algorithm

MIN takes 21 traffic links to achieve the multicast using the same example.

3.3.2 MIN Algorithm in 2D Mesh

This section presents formal description of *MIN* multicast algorithm in *2*D meshes.

Algorithm 3 ***(MIN Algorithm in 2D Mesh):*** In a *2*D mesh network of size $(m \times n)$, given a source node $s(0,0)$ and a set of k destination nodes $\{(x_1, y_1), (x_2, y_2) \dots (x_i, y_i)\}$, where $0 \leq x_i \leq m-1$, $0 \leq y_i \leq n-1$ and $i = [1, k]$. Multicasting requires message to be sent from the source node s to all k destinations.

The main steps of *MIN* multicast routing algorithm for a *2*D mesh are as follows:

Step 1. Find a pair of destination nodes A and B, where node A has the minimum x value x_{min} in the destination set and node B has the minimum y value y_{min} in the destination set.

Step 2. Construct a multicasting tree between the source node and the node (x_{min}, y_{min}).

Step 3. Connect node A and B to node (x_{min}, y_{min}).

Step 4. Select a node with minimum x value in the remaining destination set, and find a shortest path to the existing multicast tree.

Step 5. Choose a node with a minimum y value in the remaining destination set, and find the shortest path to the existing multicast tree if this node is different from the node found in *Step 4*.

Step 6. Repeat *Steps 4* and *5* until every destination is included in the multicast tree.

From the *arranged* list of destinations, it is straightforward that node with coordinates (x_1, y_1) is node A that has x_{min} value and node with coordinates (x_2, y_2) is node B that has y_{min} value. A multicast tree is constructed between the source node $s(0,0)$ and the intermediate node (x_{min}, y_{min}), and then node A and node B are connected to the intermediate node. Up to this point the *MIN* algorithm follows the multicasting approach as suggested by *PAIR* algorithm. Hereafter, *MIN* follows the multicasting approach suggested by *DIST* algorithm. It looks for the node with minimum x value and minimum y value in the remaining destination set alternatively and connects them to existing multicast tree using the shortest path.

Same example is considered for multicasting in mesh using *MIN* algorithm, as it was used for *PAIR* algorithm.

Figure 22 shows 10×9 mesh network with source node $s(0,0)$ and all destinations nodes.

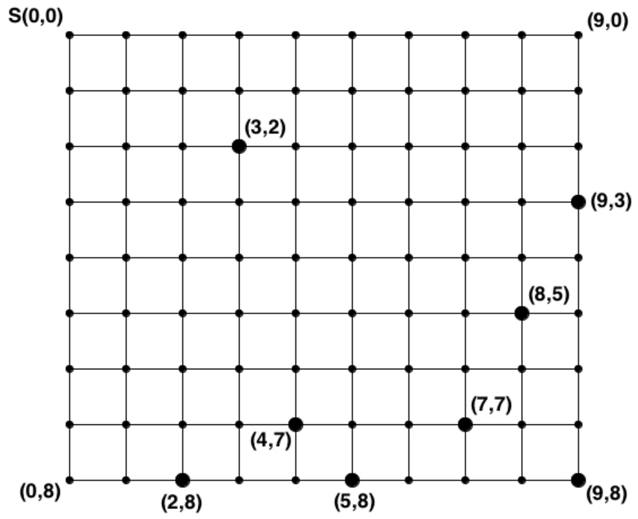

Figure 22 Mesh for MIN Algorithm

First destination nodes are *arranged*. The arranged destination set for *Example 2* is $\{(2,8),(3,2),(4,7),(9,3),(5,8),(8,5),(7,7),(9,8)\}$.

After destination nodes are arranged, pairing of first two nodes in the arranged destination set is done in order to find the intermediate node as

$\{(2,8),(3,2)\} \Rightarrow (2,2)$

Figure 23 shows the mesh network with the intermediate node (marked by **X**). The source node $s(0,0)$ is connected to the intermediate node $(2,2)$.

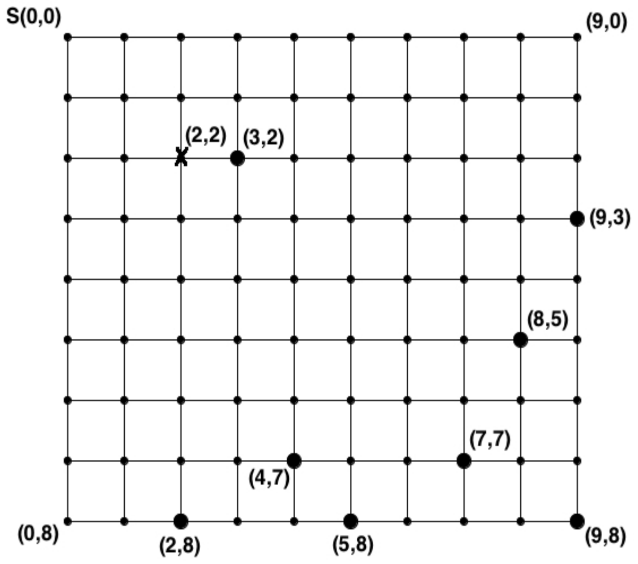

Figure 23 Intermediate Node of MIN Algorithm

When the intermediate node $(2,2)$ receives the message it forwards the message to destination nodes $(3,2)$ and $(2,8)$ as shown in *Figure 24*, and these are the first three steps of the *MIN* algorithm which are similar to *PAIR* algorithm.

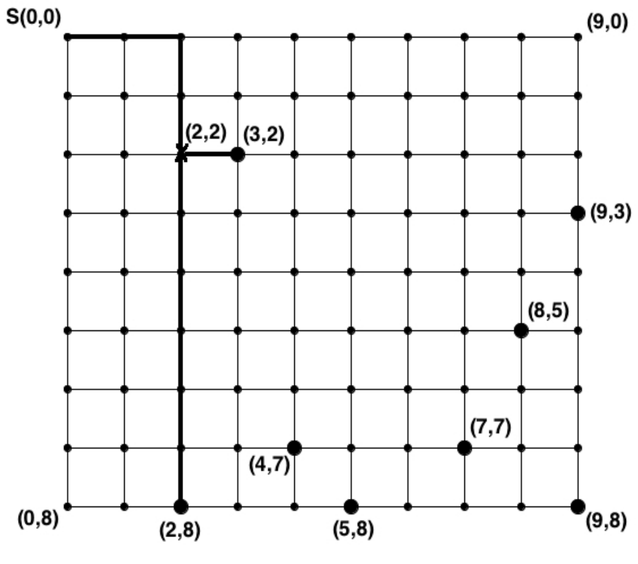

Figure 24 First three steps of MIN

Thereafter, $(4,7)$ is selected as the node that has minimum x value in the remaining destination set and is connected to the existing multicast tree. Node $(9,3)$ is the selected node that has minimum y value in the remaining destination set and is connected to existing multicast tree. This process is repeated until all destinations get connected to the existing multicast tree as shown in *Figure 25*.

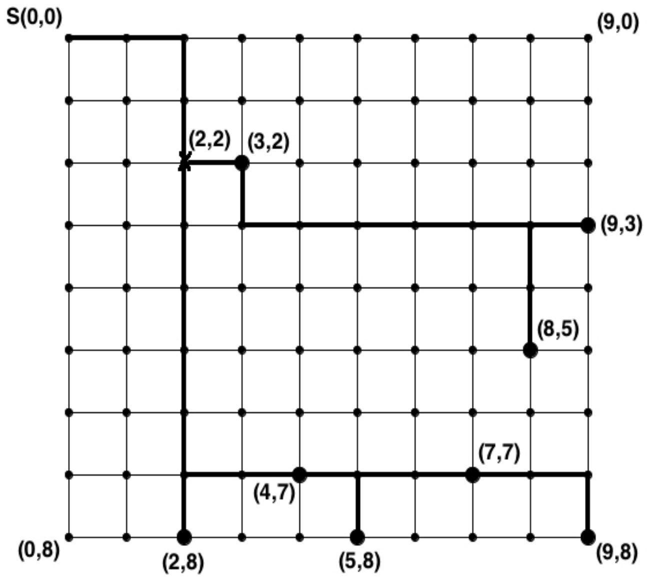

Figure 25 MIN Algorithm on Mesh of *Example 2*

Figure 25 shows complete multicasting scheme on a (10×9) mesh network with a given set of destinations. The number of links used to multicast a message from the source to all destinations is 29 and time taken to achieve this multicast is 20 units.

For the same example *PAIR* algorithm generates 34 traffic links and achieves multicast in 17 time units.

Proposition 1**:** Time complexity of *MIN* algorithm in 2D meshes is $O(kDN)$, where k is the number of destination nodes, D is the diameter of the mesh network and N is the number of nodes in the network.

Proof: Time complexity of arranging k destinations is $O(k\ logk)$. Construction of multicast tree using a shortest path takes $O(Dk(N + \sum y_i + \sum x_i)) = O(kDN)$.

Hence the overall worst-case running time of *MIN* in a 2D mesh becomes

$O(k\ logk + kDN) = O(kDN)$, as $DN > logk$.

Proposition 2: In the multicast tree constructed by *MIN* algorithm, the path from the source node to any destination node is a shortest one.

Proof: The intermediate node is connected to the source node using the shortest path, and then for reaching the first two destinations from the intermediate node again a shortest path is used. Subsequently, other destinations are connected to the existing multicast tree using a shortest path, which ultimately results in shortest path from source node to any destination node.

Proposition 3: *MIN* multicast routing algorithm is deadlock free.

The explanation is similar to that of *PAIR* multicast routing algorithm.

Proposition 4: The bound of *MIN* algorithm's multicast traffic in 2D mesh network assuming destination nodes are *arranged* is

$Traffic_{MIN} \leq (x_2 + y_1) + \sum_{j=3}^{k}(|x_j - x_{j-2}| + |y_j - y_{j-2}|)$ where k is the number of total destinations and x_2, y_1 are the coordinates of first two nodes of the *arranged* destination set.

Proof: This bound is achieved when the shortest path from the current node (x_j, y_j) to the existing tree is the shortest path from node (x_j, y_j) to node (x_{j-2}, y_{j-2}). Node A with minimum x_1 value and node B with minimum y_2 value can be reached in $(x_2 + y_1)$ number of links. Other destinations are reached using a shortest path from the existing multicast tree.

Proposition 5: The bound of *MIN* algorithm's multicast time in 2D meshes is $D_{max} \leq Time_{Pair} \leq 2D_{max} - 1$, where D_{max} is the farthest destination node.

The explanation is similar to that of *PAIR* multicast routing algorithm.

3.3.3 MIN Algorithm in 2D Torus

Similar to the *PAIR* algorithm in 2D torus, *MIN* algorithm in a 2D torus is also divided into 4 sub-meshes. Multicasting is done in 4 different sub-meshes each with their own source node.

Algorithm 4 ***(MIN Algorithm in 2D Torus):*** In a 2D torus network of size $(m \times n)$, given a source node $s(0,0)$ and a set of k destination nodes $\{(x_1, y_1), (x_2, y_2) \dots (x_i, y_i)\}$, where $0 \leq x_i \leq m - 1$, $0 \leq y_i \leq n - 1$ and $i = [1, k]$. Multicasting requires message to be sent from the source node s to all k destinations.

The main steps of *MIN* multicast routing algorithm for a 2D torus are:

Step 1. Divide the torus T into four sub-meshes M_1, M_2, M_3, M_4 as described earlier.

Step 2. Partition the destination set D into four subsets D_1, D_2, D_3, D_4, which contains the destination nodes in sub-mesh M_1, M_2, M_3, M_4 respectively.

Step 3. Assume $s_1(0,0)$ as the source node and origin of sub-mesh M_1, D_1 as the set of destination nodes, apply *MIN* algorithm in sub-mesh M_1 and get the multicast sub-tree MT_1. Similarly apply the *MIN* algorithm for M_2, M_3, M_4 to get multicast sub-tree MT_2, MT_3, MT_4.

Step 4. Assemble the multicast sub-trees MT_1, MT_2, MT_3, MT_4, into the overall multicast tree MT that is rooted at the source node.

Same example is considered for multicasting in torus using *MIN* algorithm, as it was used for *PAIR* algorithm.

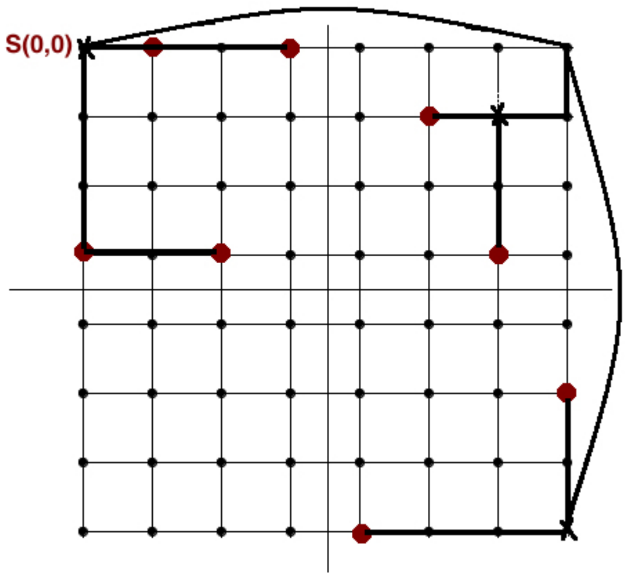

Figure 26 MIN Algorithm on Torus for *Example 3*

Figure 26 shows how multicasting is achieved in a *2*D torus using MIN algorithm. Traffic generated in this example is 20 links and time taken to achieve multicast is 6 hops.

Traffic generated by *PAIR* algorithm for the same example in a *2*D torus is 21 links.

Chapter 4

Simulation and Analysis

The purpose of this chapter is to generate and evaluate the results of *PAIR* and *MIN* algorithms in order to verify the goals of designing them. An application has been developed to simulate the multicast communication that calculates the multicast parameters, at a given size of mesh or torus network and number of destination nodes.

4.1 Implementation of Simulation System

The simulation program is developed for newly designed algorithms and some other previously published algorithms, which provides the benchmark to compare the results of multicast parameters. Object oriented programming is effective because of several advantages such as simplicity, modularity, maintainability and re-usability. Any object oriented language such as C++, Java, C# can be used to develop the application. C++ has been chosen for its efficiency in cases like this research where intensive computation is required. Specifically, a simulation application has been developed using *Visual C++* and *Microsoft Foundation Class Library* within *Microsoft Visual Studio 10.0* under *Windows 7* environment. The application can be run under any Windows operating system with *Win32* and *MFC 10.0* installed. To ensure good speed and efficiency of the application, it is recommended to use a computer system with at least *1.5 Gigahertz* CPU clock speed and *2 Gigabytes* of memory.

4.2 Performance Evaluation Model

The multicast routing algorithms presented in this work are heuristic algorithms and none of them is best for all scenarios. One algorithm performs better than the other algorithm in some cases, but worse in other cases. So, it is recommended to use the average value of a large number of many simulations run on a similar set of data using different algorithms.

The *time*, *traffic*, and *additional traffic* reflects the changing trend of the average multicast time, average multicast traffic and average additional multicast traffic with respect to the number of destination nodes. A*dditional traffic* is a better parameter to indicate the efficiency of the multicast traffic. Several graphs and tables are used for these parameters to evaluate and analyze the performance of multicast algorithms.

To compare the performance of several simulation runs in one graph, results are generated using same set of data for each run. This process of generating nodes is repeated 100 times, each algorithm is then run over the generated nodes and results are compared. Every point on the graph is averaged over many runs of multicast with the same number of destination nodes. The set of multicast destination nodes is generated randomly through a pseudo random number generator.

Another parameter used is *mean*, which indicates the average performance of a parameter using a particular algorithm. In general, *mean multicast time*, *mean multicast traffic* and *mean average additional traffic* are used as parameters to evaluate an algorithm.

4.2.1 Simulation Assumption

Following are the assumptions and conditions under which the simulations are done to evaluate the performance of algorithms. These assumptions and conditions are fairly similar to [3, 4], as the newly designed algorithms are compared with these algorithms.

- All simulations are done for *store-and-forward* switched 2-D mesh and torus networks.
- Nodes in the network have one-port architecture, which is the most commonly used architecture in multicomputers.
- To simplify the calculation, the size of both dimension of the network is assumed equal.
- The unit of time is *hops* and the unit of traffic is *number of links* or simply *links*.
- The sampling resolution for the graphs is 20 destination nodes per sample point. The value at each sample point is averaged over 100 runs of multicasts with the same number of destination nodes.
- Dimension-ordered routing is used as the routing function between pairs of nodes that results in a shortest path and prevents deadlock.
- The formulas to calculate *multicast time*, *multicast traffic* and *average additional traffic*, as defined in *Section 2.1.2.4* for *store-and-forward* switched networks with one-port architecture are:

$$\textbf{\textit{Multicast Traffic}} = |\boldsymbol{E(MT)}|$$

$$\textbf{\textit{Multicast Time}} = \boldsymbol{max\{l_1 + w_1, l_2 + w_2, \dots, l_{k-1} + w_{k-1}, l_k + w_k\}}$$

$$\boldsymbol{Additional\ Traffic = |E(MT)| - K}$$

4.2.2 Confidence Interval

In this section, a statistical concept called confidence interval is used to obtain reliable intervals of the simulation results for *PAIR* and *MIN* algorithms.

In statistics, a confidence interval (CI) is a particular kind of interval estimate of a population parameter. Instead of estimating the parameter by a single value, an interval likely to include the parameter is given. Thus, confidence intervals are used to indicate the reliability of an estimate. How likely the interval is to contain the parameter is determined by the *confidence level* or *confidence coefficient*. The confidence level would indicate the probability that the range captures the parameter. This value is represented by a percentage, so when expressed "*95% confident* that the parameter is in confidence interval", implies that 95% of the observed confidence intervals will hold the value of the parameter. Increasing the desired confidence level will widen the confidence interval. The end points of the confidence interval are referred as confidence limits. Commonly used confidence intervals are 90%, 95%, 99% [29].

For each point in the graph, simulations are run 100 times. The generated result samples(100) for each algorithm are collected, after which 95% confidence interval of the population is computed based on the mean and the standard deviation of the set. As confidence interval is based on the sample mean, it is always denoted in the form of an interval around the mean.

In this thesis, although the concept of confidence interval is used, the results of a particular sample may vary. This is due the fact that the destination nodes of the sample set are generated randomly.

4.3 Performance Evaluation of PAIR and MIN

4.3.1 Simulation Results of PAIR in 2D Mesh

The performance of *PAIR* algorithm in *2*-D mesh is studied through simulations done on a 20×20 mesh and is compared with previous time-optimal *DIAG* algorithm.

Multicast Time

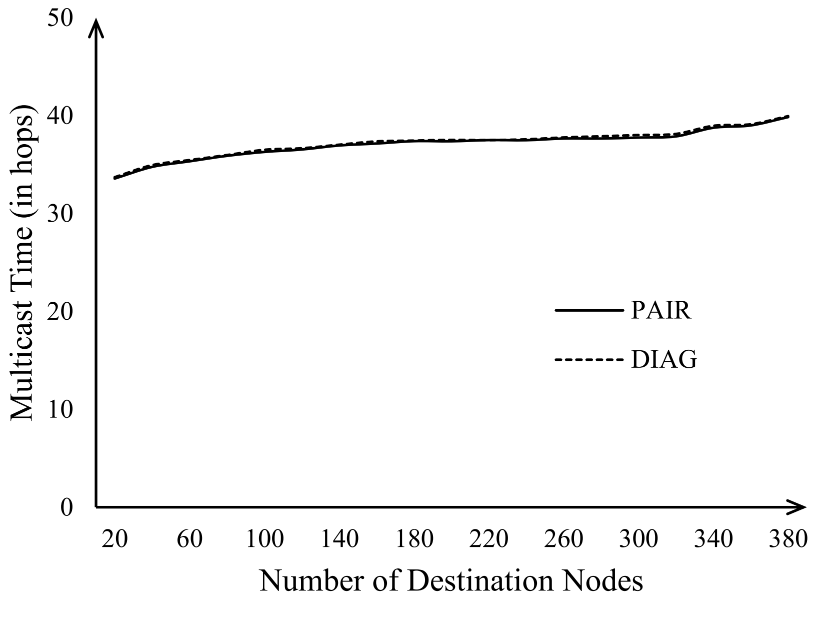

Figure 27 Multicast Time of PAIR versus DIAG in 2D Mesh

Figure 27 shows that the multicast time curves of *PAIR* and *DIAG* are almost overlapped, which indicates that the result of *mean multicast time* from *PAIR* algorithm is very similar to *mean multicast time* of *DIAG*. This is due to the fact that both *PAIR* and *DIAG* are *time optimal* multicast algorithms.

Nodes	DIAG	**PAIR**	**95% CI PAIR**	**MIN**	**95% CI MIN**
20	33.72	33.58	32.76-34.4	36.42	35.53-37.31
40	34.95	34.77	34.01-35.53	36.69	35.91-37.47
60	35.47	35.34	34.62-36.06	36.91	36.15-37.67
80	35.98	35.9	35.23-36.57	37.17	36.46-37.88
100	36.52	36.3	35.69-36.91	37.4	36.74-38.06
120	36.68	36.55	35.97-37.13	37.67	37.04-38.3
140	37.04	36.96	36.42-37.5	37.79	37.2-38.38
160	37.38	37.16	36.64-37.68	37.87	37.3-38.44
180	37.46	37.4	36.93-37.87	37.99	37.46-38.52
200	37.53	37.39	36.94-37.84	38.12	37.62-38.62
220	37.51	37.51	37.09-37.93	38.26	37.78-38.74
240	37.59	37.5	37.1-37.9	38.42	37.99-38.85
260	37.78	37.67	37.29-38.05	38.61	38.21-39.01
280	37.9	37.67	37.35-37.99	38.7	38.33-39.07
300	38.03	37.77	37.48-38.06	38.77	38.43-39.11
320	38.14	37.89	37.63-38.15	38.86	38.56-39.16
340	38.97	38.77	38.56-38.98	38.99	38.72-39.26
360	39.13	39.01	38.85-39.17	39.58	39.33-39.83
380	39.96	39.86	39.74-39.98	39.98	39.77-40.19

Table 1 Multicast Time in 2D Mesh

Individual value of a point in the above graph is extracted from *Table 1*. It is clear that the *multicast time* curve is pretty flat and it does not relate very much to the number of destinations. This happens because the *multicast time* is mainly dependent on the distance from the source node to the farthest destination node and not on the number of destination nodes. However it usually increases with the increase in number of destinations. In the above table, the mean multicast time of *PAIR* is *37.10* hops, which is *0.40%* better than *DIAG's 37.25* hops.

Multicast Traffic

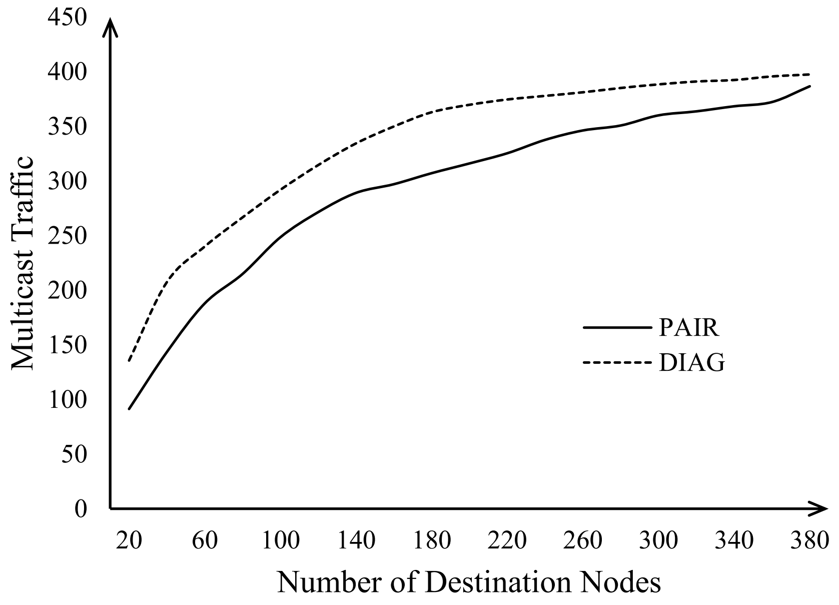

Figure 28 Multicast Traffic of PAIR versus DIAG in 2D Mesh

Figure 28 suggests that *multicast traffic* generally increases with the number of destinations and it can never be less than number of destinations. The graph also indicates that on average *PAIR* algorithm generates less traffic than *DIAG* algorithm.

Nodes	DIAG	**PAIR**	**95% CI PAIR**	**MIN**	**95% CI MIN**
20	135.6	91.28	89.85-92.71	75.98	74.82-77.14
40	207.37	143.42	142.17-144.67	95.48	94-96.96
60	239.9	187.83	186.55-189.11	111.02	109.93-112.11
80	266.54	214.78	213.27-216.29	135.71	134.23-137.19
100	291.79	248.35	247.18-249.52	161.03	159.95-162.11
120	314.41	271.35	269.04-273.66	187.58	186.26-188.9
140	334.27	288.98	287-290.96	212.93	211.47-214.39
160	349.76	296.92	295.46-298.38	237.39	235.75-239.03
180	362.79	307.01	304.97-309.05	260.19	257.54-262.84
200	369.61	315.94	313.27-318.61	278.04	276.23-279.85
220	374.46	325.26	323.15-327.37	290.99	289.26-292.72
240	377.76	337.48	335.61-339.35	298.1	296.27-299.93
260	381.05	346.11	344.35-347.87	305.3	303.32-307.28
280	384.99	350.65	348.66-352.64	316.88	314.85-318.91
300	388.3	359.94	357.93-361.95	325.03	322.71-327.35
320	390.92	363.6	361.55-365.65	337.04	335.16-338.92
340	392.3	368.25	366.07-370.43	350.48	348.72-352.24
360	395.56	372.13	370.16-374.1	365.71	363.79-367.63
380	397.37	386.55	384.68-388.42	382.06	380.19-383.93

Table 2 Multicast Traffic in 2D Mesh

Table 2 shows *multicast traffic* in *2*-D mesh. Multicast traffic varies depending on the distribution of the destination nodes. If all the destination nodes are located closer to the source node, multicast traffic will be less as compared to when all the destination nodes are located at a relatively farther distance. In the above table the mean multicast traffic of *PAIR* algorithm is *293.46* links, which is *12%* less than *DIAG's 334.06* links.

Thus *PAIR* algorithm performs better on *multicast traffic* than *DIAG* algorithm in most cases.

Average Additional Traffic

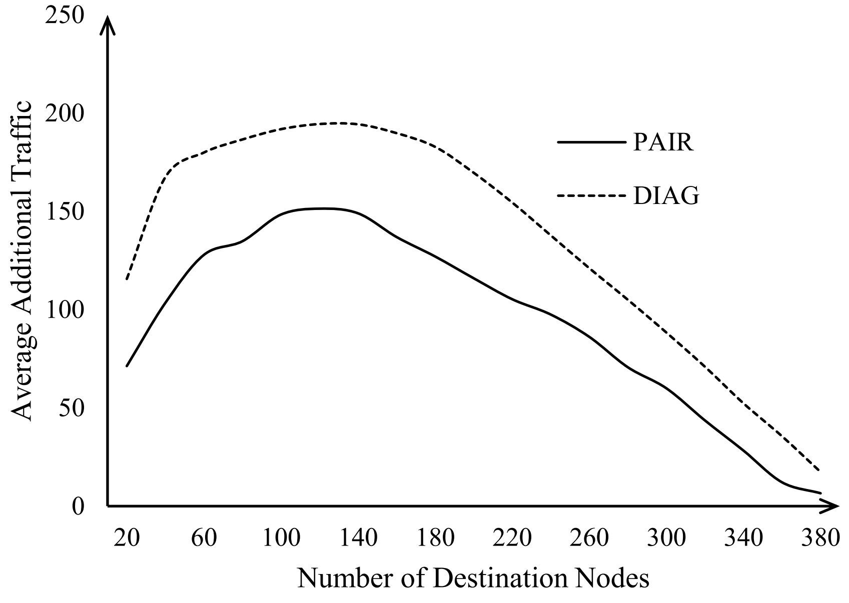

Figure 29 AAT of PAIR versus DIAG in 2D Mesh

Average additional traffic (AAT) first increases with the number of destinations and reaches at its peak at about 30% of the total nodes, and then decreases as the number of destination reaches their maximum value. This implies that when the number of destination nodes is very small or very large compared to the total nodes in the network, the overall *multicast traffic* is most efficient. In between, total *multicast traffic* generated will be more. *Figure 29* indicates that performance of *PAIR* algorithm is better than that of *DIAG* algorithm, which means that PAIR algorithm generates less *average additional traffic* for a given number of destination nodes.

Nodes	DIAG	**PAIR**	**95% CI PAIR**	**MIN**	**95% CI PAIR**
20	115.6	71.28	69.87-72.69	55.98	54.87-57.09
40	167.37	103.42	102.21-104.63	55.48	54.05-56.91
60	179.9	127.83	126.59-129.07	51.02	49.98-52.06
80	186.54	134.78	133.32-136.24	55.71	54.26-57.16
100	191.79	148.35	147.22-149.48	61.03	60.01-62.05
120	194.41	151.35	149.11-153.59	67.58	66.33-68.83
140	194.27	148.98	147.07-150.89	72.93	71.51-74.35
160	189.76	136.92	135.51-138.33	77.39	75.78-79
180	182.79	127.01	125.04-128.98	80.19	77.56-82.82
200	169.61	115.94	113.33-118.55	78.04	76.23-79.85
220	154.46	105.26	103.19-107.33	70.99	69.31-72.67
240	137.76	97.48	95.65-99.31	58.1	56.3-59.9
260	121.05	86.11	84.42-87.8	45.3	43.38-47.22
280	104.99	70.65	68.72-72.58	36.88	34.89-38.87
300	88.3	59.94	58.05-61.83	25.03	22.74-27.32
320	70.92	43.6	41.66-45.54	17.04	15.21-18.87
340	52.3	28.25	26.13-30.37	10.48	8.74-12.22
360	35.56	12.13	10.22-14.04	5.71	3.88-7.54
380	17.37	6.55	4.73-8.37	2.06	0.21-3.91

Table 3 AAT in 2D Mesh

Table 3 shows that for a mesh of size 20×20, *average additional traffic* increases till the number of destination nodes become *120* (*30%*) and then it starts decreasing. This is due to the fact that ratio of number of links and destination nodes gets closer to 1, hence additional links used will reach its minimal value. From *Table 3*, it is clear that *mean average additional traffic* of *PAIR* algorithm performs better than that of *DIAG* algorithm by over *30%*. The *mean* of *PAIR* algorithm's *average additional traffic* is *93.46* whereas *mean* of *DIAG's average additional traffic* is *134.46*.

From the simulation results of *PAIR* and *DIAG* algorithms, it has been determined that *PAIR* algorithm reduces the *multicast traffic* by *12%* and increases the traffic efficiency by *30%* as compared to *DIAG* algorithm, without increasing the optimal *multicast time*.

4.3.2 Simulation Results of PAIR in 2D Torus

The performance of *PAIR* algorithm in *2*-D torus is studied through simulations done on a 20×20 torus and is compared with previous time-optimal *DIAG* algorithm.

Multicast Time

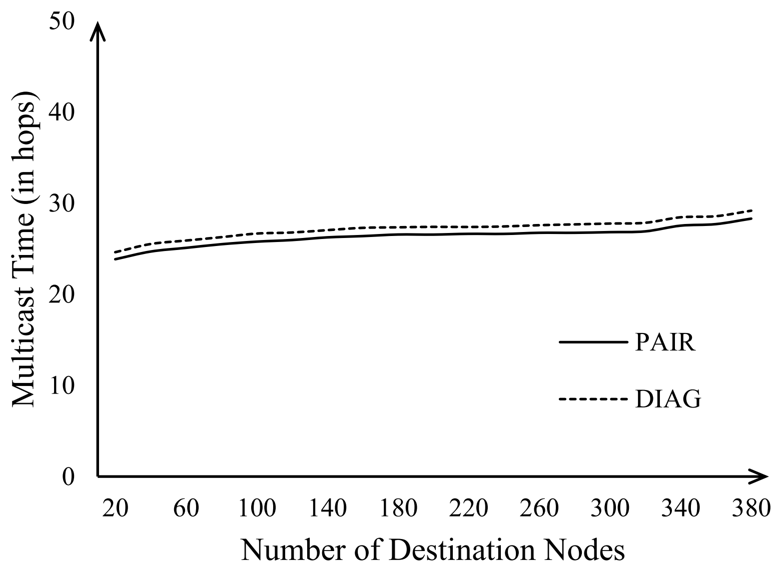

Figure 30 Multicast Time of PAIR versus DIAG in 2D Torus

Figure 30 shows that there is very little difference in the multicast time curves of *PAIR* and *DIAG* algorithms, which implies that the results of *mean multicast time* from *PAIR* algorithm is very close to the *mean multicast time* of *DIAG* for a *2*-D torus.

Nodes	DIAG	**PAIR**	**95% CI PAIR**	**MIN**	**95% CI MIN**
20	24.62	23.84	23.46-24.22	25.13	24.38-25.88
40	25.51	24.69	24.16-25.22	25.32	25.08-25.56
60	25.89	25.09	24.44-25.74	25.47	24.95-25.99
80	26.27	25.49	25.37-25.61	25.65	24.86-26.44
100	26.66	25.77	25.19-26.35	25.81	25.68-25.94
120	26.78	25.95	25.17-26.73	25.99	25.37-26.61
140	27.04	26.24	25.54-26.94	26.37	25.89-26.85
160	27.29	26.38	25.62-27.14	26.49	25.98-27
180	27.35	26.55	26.21-26.89	26.89	26.45-27.33
200	27.40	26.55	26.21-26.89	26.97	26.65-27.29
220	27.38	26.63	25.99-27.27	27.16	26.94-27.38
240	27.44	26.63	26.18-27.08	27.22	26.84-27.6
260	27.58	26.75	26.54-26.96	27.38	27.09-27.67
280	27.67	26.75	25.96-27.54	27.60	26.89-28.31
300	27.76	26.82	26.14-27.5	27.79	27.37-28.21
320	27.84	26.90	26.16-27.64	27.93	27.35-28.51
340	28.45	27.53	27.32-27.74	28.69	27.91-29.47
360	28.56	27.70	27.28-28.12	28.97	28.69-29.25
380	29.17	28.30	27.68-28.92	29.29	28.55-30.03

Table 4 Multicast Time in 2D Torus

Similar to *2*-D mesh, it is clear that the *multicast time* curve is pretty flat and this implies that it is not very much related to the number of destinations. The *multicast time* of a *2*-D torus is less than that of a *2*-D mesh, it is reasonable as the diameter of a torus is half of a respective mesh. In the above table, the *mean multicast time* of *PAIR* algorithm for *2*-D torus network is *26.34* hops, which is *3.12%* better than *DIAG's 27.19* hops.

Multicast Traffic

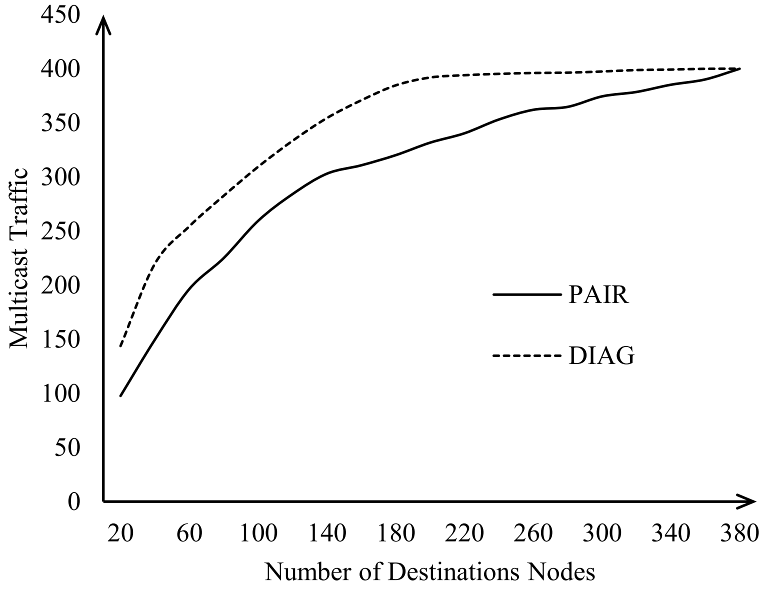

Figure 31 Multicast Traffic of PAIR versus DIAG for 2D Torus

Figure 31 shows *multicast traffic* of *PAIR* versus *DIAG* for a *2*-D torus. The graph also indicates that on an average *PAIR* algorithm generates less *multicast traffic* than *DIAG* algorithm.

Nodes	DIAG	**PAIR**	**95% CI PAIR**	**MIN**	**95% CI MIN**
20	143.73	97.63	96.16-99.1	79.78	78.59-80.97
40	219.81	149.68	148.37-150.99	100.25	98.76-101.74
60	254.29	196.36	195.02-197.7	121.57	120.42-122.72
80	282.53	224.97	223.43-226.51	142.50	140.99-144.01
100	309.29	259.34	258.11-260.57	169.08	167.96-170.2
120	333.27	284.04	281.69-286.39	196.96	195.58-198.34
140	354.32	302.92	300.88-304.96	223.58	222.09-225.07
160	370.74	310.68	309.19-312.17	249.26	247.58-250.94
180	384.55	320.04	317.96-322.12	273.20	270.51-275.89
200	391.78	331.59	328.88-334.3	291.94	290.11-293.77
220	393.92	340.27	338.11-342.43	305.54	303.76-307.32
240	395.16	352.97	351.02-354.92	313.01	311.15-314.87
260	395.97	361.95	360.12-363.78	320.57	318.48-322.66
280	396.28	364.67	362.64-366.7	332.72	330.65-334.79
300	397.34	374.33	372.25-376.41	341.28	338.92-343.64
320	398.67	378.44	376.31-380.57	353.89	351.95-355.83
340	399.12	385.11	382.85-387.37	368.00	366.21-369.79
360	399.83	390.01	387.96-392.06	384.00	382.02-385.98
380	399.97	399.87	397.93-401.81	397.67	395.73-399.61

Table 5 Multicast Traffic in 2D Torus

Table 5 shows that the *multicast traffic* increases with the increase in number of destination nodes. If all the destination nodes are located closer to the source node in all of the 4 sub-meshes network, *multicast traffic* will be less as compared to when the destination nodes are located at a relatively farther distance or in between of the torus network. In the above table the *mean multicast traffic* of *PAIR* algorithm in *2*-D torus network is *306.57* links, which is *12%* less than *DIAG's 348.45* links.

Average Additional Traffic

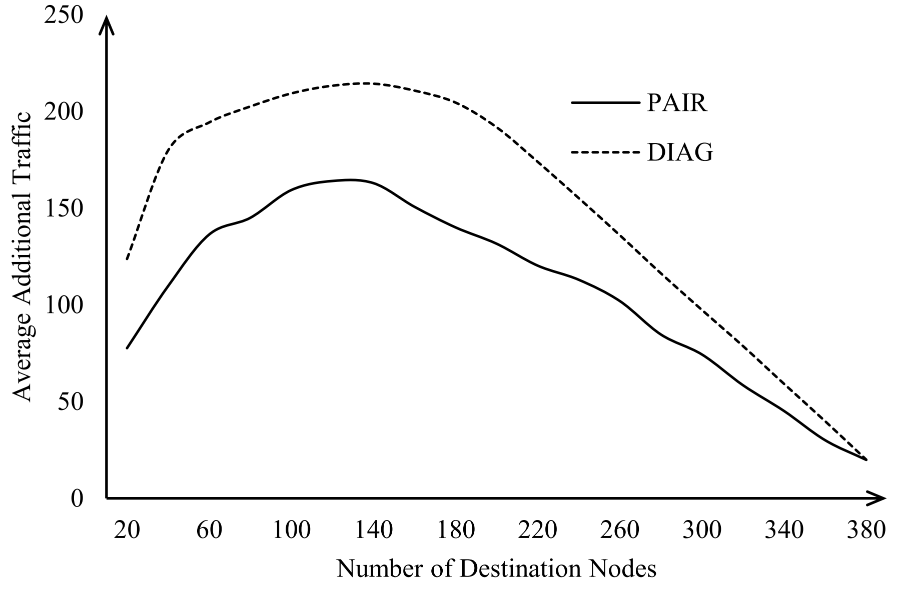

Figure 32 AAT of PAIR versus DIAG in 2D Torus

Average additional traffic (AAT) first increases with the number of destinations and reaches its peak at about *30-35%* of the total nodes, and then decreases as the number of destination reaches their maximum possible value. In between, total *multicast traffic* generated will be more.

Figure 32 indicates that performance of *PAIR* algorithm is better than that of *DIAG* algorithm as *PAIR* algorithm generates less *average additional traffic* for a given number of destination nodes.

Nodes	DIAG	**PAIR**	**95% CI PAIR**	**MIN**	**95% CI MIN**
20	123.73	77.63	76.19-79.07	59.78	58.75-60.81
40	179.81	109.68	108.43-110.93	60.25	58.87-61.63
60	194.29	136.36	135.05-137.67	61.57	60.56-62.58
80	202.53	144.97	143.44-146.5	62.50	61.08-63.92
100	209.29	159.34	158.13-160.55	69.08	68.11-70.05
120	213.27	164.04	161.72-166.36	76.96	75.76-78.16
140	214.32	162.92	160.91-164.93	83.58	82.22-84.94
160	210.74	150.68	149.24-152.12	89.26	87.74-90.78
180	204.55	140.04	138-142.08	93.20	90.62-95.78
200	191.78	131.59	128.91-134.27	91.94	90.16-93.72
220	173.92	120.27	118.15-122.39	85.54	83.89-87.19
240	155.16	112.97	111.06-114.88	73.01	71.25-74.77
260	135.97	101.95	100.19-103.71	60.56	58.67-62.45
280	116.28	84.67	82.71-86.63	52.72	50.76-54.68
300	97.34	74.33	72.29-76.37	41.28	39.04-43.52
320	78.67	58.44	56.36-60.52	33.89	32.1-35.68
340	59.12	45.11	42.93-47.29	28.00	26.32-29.68
360	39.83	30.01	27.99-32.03	24.00	22.22-25.78
380	19.97	19.87	17.96-21.78	17.67	15.86-19.48

Table 6 AAT in 2D Torus

Table 6 shows that in a torus of size 20×20, average additional traffic increases till the number of destination nodes become *120-140* (30-35%) and then it starts decreasing. From *Table 6,* it is clear that the *mean average additional traffic* of *PAIR* algorithm performs better than that of *DIAG* algorithm by over *28%*. The mean of *PAIR* algorithm's average additional traffic is 106.57 whereas the mean of *DIAG's* average additional traffic is *148.45*.

From the simulation results of *PAIR* and *DIAG* in *2*-D torus*,* it has been determined that *PAIR* algorithm reduces the *multicast traffic* by 12% and increases the traffic efficiency by *28*% over *DIAG* algorithm, but does not increase the optimal *multicast time*.

4.3.3 Simulation Results of MIN in 2D Mesh

The performance of *MIN* algorithm in *2*-D mesh is studied through simulations done on a 20×20 mesh and is compared with *PAIR* algorithm, which generates less *multicast traffic* than any other time optimal algorithm.

Multicast Time

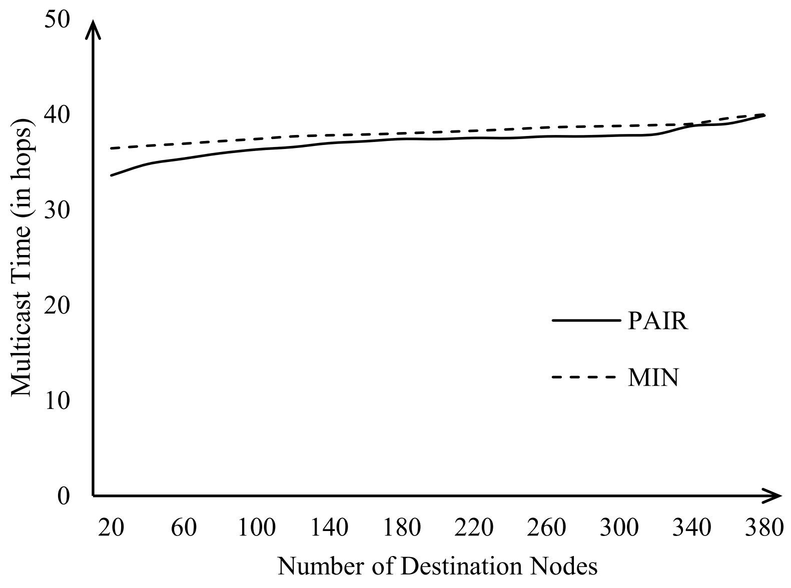

Figure 33 Multicast Time of MIN versus PAIR in 2D Mesh

Figure 33 shows the relation between *multicast time* of *MIN* and *PAIR* algorithm. *MIN* algorithm does not perform as well as *PAIR* in terms of *multicast time*. This is because of the fact that *MIN's* main aim is to reduce the *multicast traffic* while achieving a near optimal *multicast time*.

PAIR algorithm's *mean multicast time* is *37.10* hops, which is over *2.5%* better than that of MIN's *38.12* hops.

The data for *Figure 33* can be extracted from *Table 1*.

Multicast Traffic

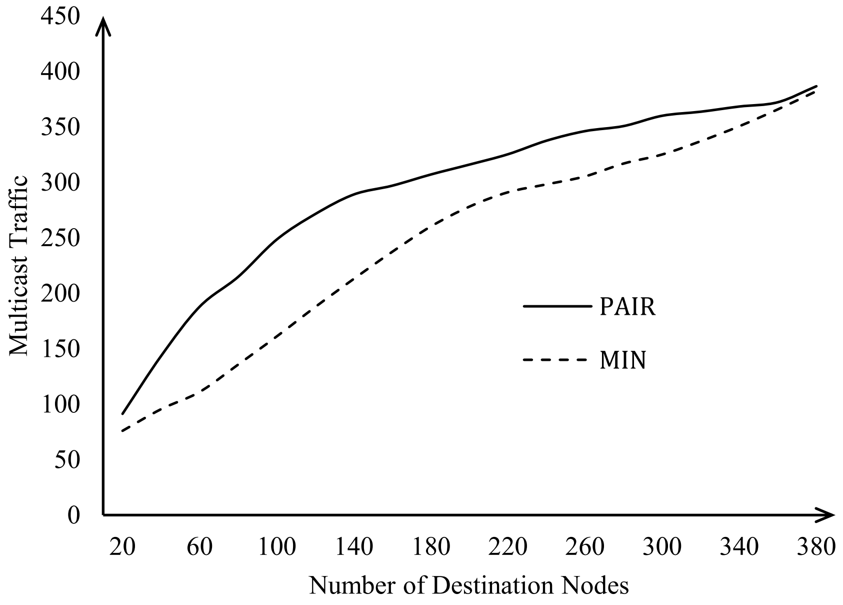

Figure 34 Mutlicast Traffic of MIN versus PAIR in 2D Mesh

Figure 34 indicates that on average *MIN* algorithm generates less traffic than *PAIR* algorithm. The *mean multicast traffic* of *MIN* algorithm is *248.79* links, which is *15%*

less than *PAIR's 293.46* links. Thus *MIN* algorithm performs better on *multicast traffic* than *PAIR* algorithm in most cases.

The data for *Figure 34* can be extracted from *Table 2*.

Average Additional Traffic

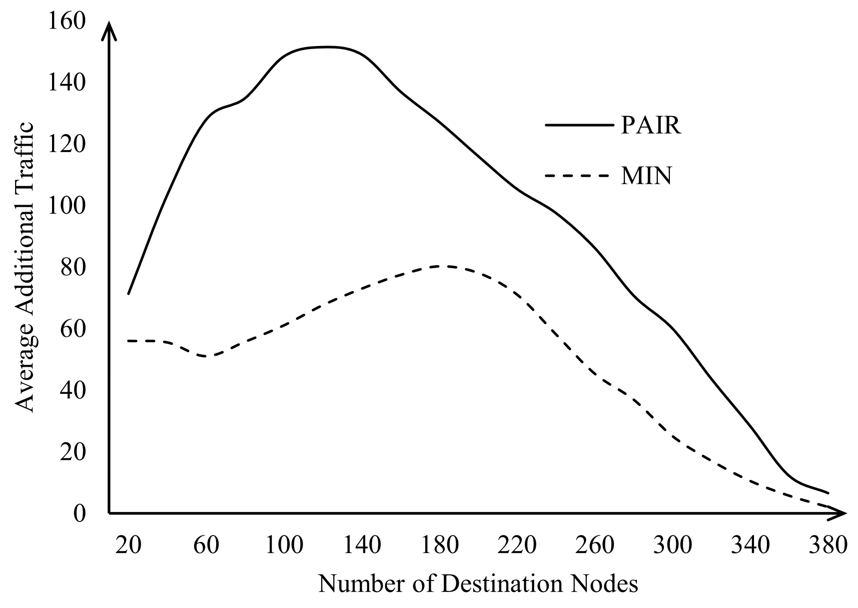

Figure 35 AAT of MIN versus PAIR in 2D Mesh

Average additional traffic (AAT) first increases with the number of destinations and reaches its peak at about *35-45%* of the total nodes. *Mean average additional traffic* of *MIN* algorithm is better than that of *PAIR* algorithm by about *48%*. The mean of *MIN* algorithm's *average additional traffic is 48.78* whereas mean of *PAIR's average additional traffic* is *93.46*.

The data for *Figure 35* can be extracted from *Table 3*.

From the simulation results of *PAIR* and *MIN* on a *2*-D mesh, it has been determined that *MIN* algorithm reduces the *multicast traffic* by *15%* and increases the traffic efficiency by *48% PAIR* algorithm, with a little increase in the optimal *multicast time*.

4.3.4 Simulation Results of MIN in 2D Torus

The performance of MIN algorithm in *2*-D torus is studied through simulations done on a 20×20 torus and is compared with *PAIR* algorithm, which generates less *multicast traffic* than any other time optimal algorithm.

Multicast Time

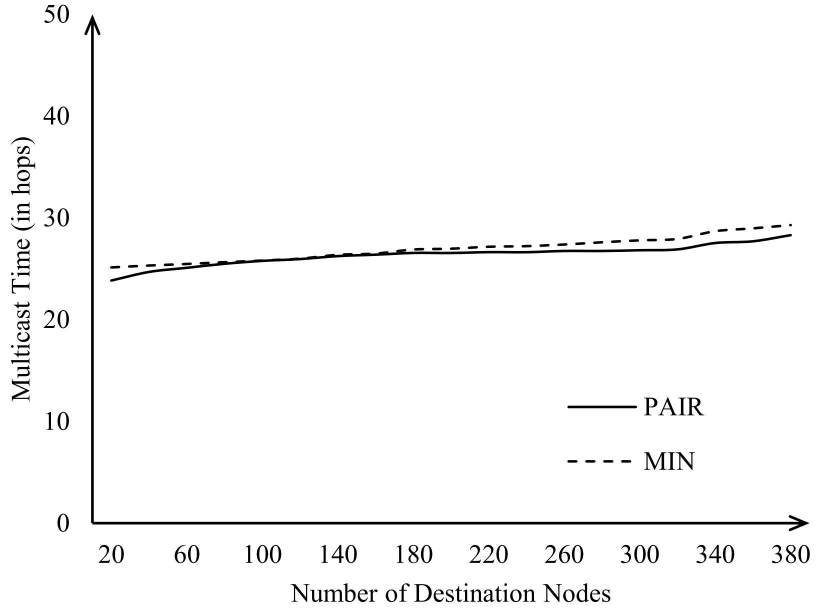

Figure 36 Multicast Time of MIN versus PAIR in 2D Torus

Figure 36 shows the relation between *multicast time* of *MIN* and *PAIR* algorithm. *PAIR* algorithm's *mean multicast time* is *26.34* hops, which is over *2%* better than that of *MIN's 26.95* hops.

The data for *Figure 36* can be extracted from *Table 4*.

Multicast Traffic

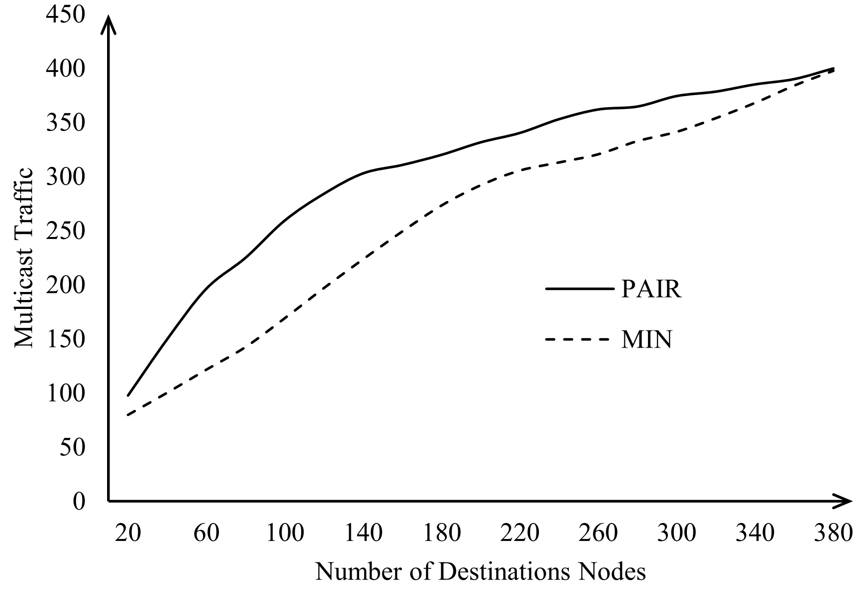

Figure 37 Multicast Traffic of MIN versus PAIR in 2D Torus

Figure 37 indicates that on an average *MIN* algorithm generates less traffic than *PAIR* algorithm. The *mean multicast traffic* of *MIN* algorithm is *261.30* links, which is about *15%* better than *PAIR's 306.57* links.

The data for *Figure 37* can be extracted from *Table 5*.

Average Additional Traffic

Average additional traffic (AAT) first increases with the number of destinations and reaches its peak at about *35-45%* of the total nodes as suggested by *Figure 38*. *Mean average additional traffic* of *MIN* algorithm is better than that of *PAIR* algorithm by over 42%. The mean of *MIN* algorithm's *average additional traffic is 61.30* links whereas mean of *PAIR's average additional traffic* is *106.57* links.

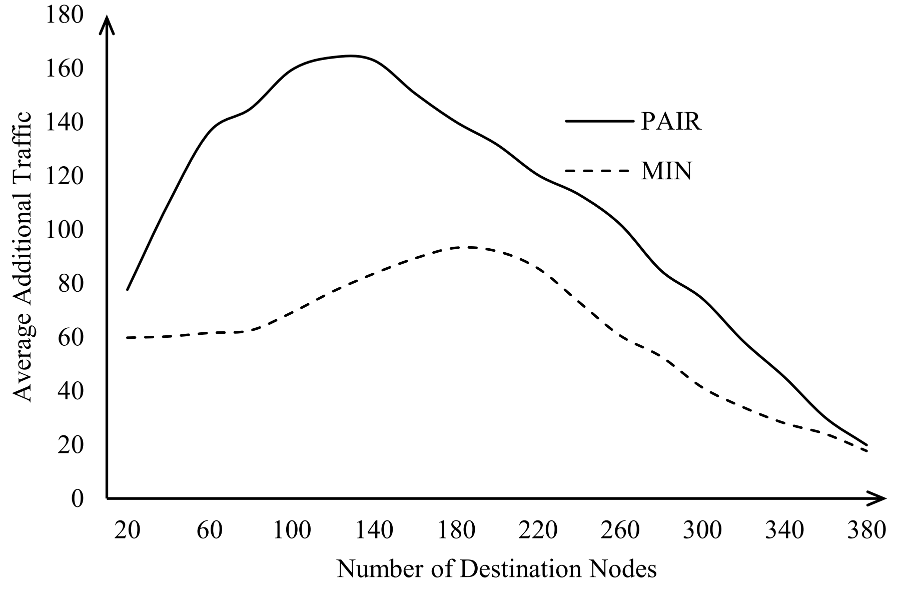

Figure 38 AAT of MIN versus PAIR in 2D Torus

The data for *Figure 38* can be extracted from *Table 6*.

From the simulation results of *PAIR* and *MIN* on *2*-D torus, it has been determined that *MIN* algorithm reduces the *multicast traffic* by *15%* and it increases the traffic efficiency by *42%* over *PAIR* algorithm, with a little increase in the optimal *multicast time* of *PAIR* algorithm.

Chapter 5

Conclusion and Future Work

Efficient multicast routing algorithms with good heuristics that can minimize both *time* and *traffic* are required to improve the performance of mesh and torus connected multicomputers under different situations.

Each algorithm designed during this research is primarily focused on one parameter first, and then tries to reduce the value of other parameter as much as possible. *PAIR* algorithm significantly reduces the *traffic* of previous time-optimal *DIAG* algorithm without an increase in the *multicast time* or time complexity of the algorithm, $O(kN)$. *MIN* algorithm further reduces the *multicast traffic* of *PAIR* algorithm but suffers in terms of *multicast time* and time complexity of the algorithm, which is $O(kDN)$ in case of *MIN*.

To tackle the challenge of NP-complete multicast problems in mesh and torus connected networks, this research was performed and the following results and conclusions are achieved.

- A time optimal tree based shortest path multicast routing algorithm called *PAIR* for store-and-forward switched mesh and torus networks has been designed that significantly increases the traffic efficiency by *30%* in *2*-D meshes and *28%* in *2*-D tori as compared to the earlier time optimal *DIAG* algorithm without sacrificing the optimal multicast time and complexity.

- Another tree based shortest path multicast routing algorithm called *MIN* for store-and-forward switched mesh and torus networks has been designed that achieves near optimal *multicast time* and significantly increases the traffic efficiency by *48%* in *2*-D meshes and *42%* in *2*-D tori over *PAIR*.
- A program to simulate multicasting in mesh and torus connected multicomputers has been designed and implemented for performance analysis, which can be easily customized for any size of mesh or torus.
- The performance of *PAIR* and *MIN* algorithms has been analyzed by comparing the simulation results with the existing algorithms in *2*-D meshes and tori networks.

Experiments and simulations in this work show that each of these algorithms has its pros and cons. None of them performs best in all cases. Therefore, more consideration and research needs to be directed in this field to come up with some more good algorithms, which can perform the best in all the cases.

The following is some of the work that can be done in future to improve the performance of these algorithms.

- Simulate these algorithms in higher dimensional meshes and tori to evaluate their performance with respect to the size and dimension.

- Develop algorithms optimized for concurrent multiple multicasting of messages.
- Integrate these multicast algorithms with deadlock prevention, fault tolerant and traffic balancing approaches and get them ready for real applications.

Bibliography

[1] A. Y. Al-Dubai, M. Ould-Khaoua and L. M. Mackenzie, *An Efficient Path-Based Multicast Algorithm for Mesh Networks*, IEEE Parallel and Distributed Processing Symposium, pp. 22-26, April 2003.

[2] E. Fleury and P. Fraigniaud, *Analysis of deadlock-free path-based wormhole multicasting in meshes in case of contentions*, Proceedings of Sixth Symposium on the Frontiers of Massively Parallel Computing, pp. 34-41, 1996.

[3] H. A. Harutyunyan and X. Liu, *New Multicast Algorithms in Mesh-connected networks,* International Symposium on Performance Evaluation of Computer and Telecommunication Systems, pp. 284- 291, 2003.

[4] H. A. Harutyunyan and S. Wang, *Efficient Multicast Algorithms for Mesh connected Multicomputers*, IEEE Tenth International Conference on Information Visualization, pp. 504-510, July 2006.

[5] C. Hubsch and O. P. Waldhorst, *Flexible tree-based Application-Layer Multicast*, 17th IEEE International Conference on Networks, pp. 159-164, 2011.

[6] W. Jia, W. Tu and J. Wu, *Hierarchical Multicast Tree Algorithms for Application Layer Mesh Networks*, International Conference on Networking and Mobile Computing, pp. 549-559, 2005.

[7] S. Jin, Y. Zhuang, L. Liu, and J. Wu, *An efficient overlay multicast routing algorithm for real-time multimedia applications*, Advances in Data and Web Management, Springer Berlin / Heidelberg, Vol. 4505, pp 829–836, 2007.

[8] G. Kuperman, E. Modiano and A. Narula-Tam, *Analysis and Algorithms for Partial Protection in Mesh Networks*, IEEE Infocom , pp. 516-520, 2011.

[9] X. Lin and L. M. Ni, *Multicast Communication in Multicomputer Networks,* IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 10, pp. 1105-1117, October 1993.

[10] X. Lin and L. M. Ni, *Deadlock-free multicast wormhole routing in multicomputer networks*, International Symposium on Computer Architecture, Volume 19, Issue 3, pp. 116-125, May 1991.

[11] Z. Liul and J. Duato, *Adaptive unicast and multicast in 3D mesh networks*, IEEE Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, pp. 173-182, January 1994.

[12] M. P. Malumbres and J. Duato, *An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors*, Journal of Systems Architecture, Volume 46 Issue 11, pp. 1019-1032, September 2000.

[13] P. K. McKinley, H. Xu, A. Esfahanian and L. M. Ni, *Unicast-Based Multicast Communication in Wormhole-Routed Networks*, IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 12, pp. 1252-1265, December 1994.

[14] P. V. Meighem, G. Hooghiemstra and R.V. Hofstad, *On the Efficiency of Multicast*, IEEE/ACM Transactions on Networking, Volume 9 Issue 6, pp. 719-732, December 2001.

[15] P. Mohapatra, *Wormhole Routing Techniques for Directly Connected Multicomputer Systems*, ACM Computing Surveys, Volume 30 Issue 3, pp. 374-410, September 1998.

[16] L. M. Ni and P. K. McKinley, *A Survey of Wormhole Routing Techniques in Direct Networks*, IEEE Computer 26(2), pp. 62-76, 1993.

[17] A. Obaid and W. Zuo, *An Efficient Path-Based Multicast Algorithm for Minimum Communication Steps*, Information Technology Journal 7, pp. 32-39, 2008.

[18] C. Y. Oh, J. Park, J. Ahn, M. Lee, T. J. Lee, W. Cha, and P. S. Mah, *Tree-based multicast protocol using multi-point relays for mobile ad hoc networks*, Third International Conference on Ubiquitous and Future Networks, pp. 174-178, 2011.

[19] J. Park, H. Choi, N. Nupairoj and L. M. Ni, *Construction of Optimal Multicast Trees Based on the Parameterized Communication Model*, International Conference on Parallel Processing, pp. 180-187, 1996.

[20] D. F. Robinson, P. K. McKinley and B. H. C. Cheng, *Optimal Multicast Communication in Wormhole-Routed Torus Networks*, IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 10, pp. 1029-1042, October 1995.

[21] F. A. Samman, T. Hollstein and M. Glesner, *Adaptive and Deadlock-Free Tree-Based Multicast Routing for Networks-on-Chip*, IEEE Transactions on Very Large Scale Integration Systems, Vol. 18, Issue 7, pp. 1067-1080, July 2010.

[22] A. S. Tanenbaum, *Computer Networks*, Fourth Edition, ISBN 0130661023, Prentice Hall, 2003.

[23] University of Wisconsin, CS838: Topics in parallel computing, http://pages.cs.wisc.edu/~tvrdik/7/html/Section7.html, Last modified: January 23.

[24] V. Venkataraman, K. Yoshida, and P. Francis, *Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast*, Proceedings of the 14th IEEE International Conference on Network Protocols, pp. 2-11, 2006.

[25] N. C. Wang, *Power-aware dual-tree-based multicast routing protocol for mobile ad hoc networks*, IET Communications, Vol. 6, Issue 7, pp. 724-732, 2012.

[26] H. Wang and D. M. Blough, *Tree-Based Fault-Tolerant Multicast in Multicomputer Networks*, Modeling, Proceedings. Sixth International Symposium on Analysis and Simulation of Computer and Telecommunication Systems, pp. 44-49, 1998.

[27] B. Wang and S. K. S. Gupta, *On maximizing lifetime of multicast trees in wireless ad hoc networks*, Proceedings International Conference on Parallel Processing, pp. 333-340, 2003.

[28] Wikimedia Foundation, *Wikipedia-The Free Encyclopedia Network Topology*, http://en.wikipedia.org/wiki/Network_topology, As seen on October 15, 2012 at 01:43.

[29] Wikimedia Foundation, *Wikipedia-The Free Encyclopedia Network Topology*, http://en.wikipedia.org/wiki/Confidence_interval, As seen on January 14, 2013 at 10:45.

[30] H. Xu, X. Wu, H. R. Sadjadpour, and J. J. Garcia-Luna-Aceves, *A Unified Analysis of Routing Protocols in MANETs*, IEEE Transactions on Communications, Vol. 58, No. 3, pp. 911-922, 2010.

[31] J. S. Yang and C. T. King, *Efficient Tree-based Multicast in Wormhole-routed 2D Meshes*, Proceedings of Third International Symposium on Parallel Architectures, Algorithms, and Networks, pp. 494-500, 1997.

[32] Z. Zhang, A. Greiner and S. Taktak, *A Reconfigurable Routing Algorithm for a Fault-Tolerant 2D-Mesh Network-on-Chip*, Design Automation Conference, pp. 441-446, 2008.