MULTICASTING ALGORITHMS FOR MESH AND TORUS NETWORKS

XIAOLIN LIU

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

JANUARY 2003

0-612-77914-9

Abstract

Multicasting Algorithms for Mesh and Torus Networks

Xiaolin Liu

Multicasting is an important interprocessor communication pattern existing in various parallel application algorithms. Mesh-connected topology is one of the most thoroughly investigated network topologies for parallel processing. The torus network has been proposed for metropolitan area networks (MAN). It can be divided into several mesh problems.

Time and traffic are the main parameters considered in the multicasting communication environment. It is NP-complete in general to find an optimal multicasting algorithm which minimizes both time and traffic. This thesis proposes two kind of multicasting algorithms for torus/mesh networks, the VH algorithm with a time complexity of O(kD), and the DIST algorithm with a time complexity of O(kDN). where k is the number of destination nodes. D is the maximum distance, and N is the total number of nodes in the network. The VH algorithm guarantee that every destination node can receive the message from the source in a minimum multicasting time. The DIST algorithm generates less traffic compared to the VH algorithm, but at the price of an increased multicasting time.

To My Lovely Parents

Mrs. Yaojun Bao and Mr. Jicheng Liu

And My Wonderful Wife and Son

Xiuwei Wang and Sihan Liu

Acknowledgement

First, I would like to thank my supervisor Dr. Hovhannes Harutyunyan who has given me support and help throughout this thesis. He always led me in the right direction whenever I felt lost. I have learned a lot from him, both for my career and my personal life. I would also like to thank all the professors at the Department of Computer Science of Concordia University for developing my knowledge, which allows me to understand the physics and mathematics behind real world problems.

I would like to thank my parents, Mrs. Yaojun Bao and Mr. Jicheng Liu for their endless love, support and encouragement since my childhood. Their financial and mental supports have been the reasons behind all my achievements.

I would like to thank my soul mate, my beautiful wife, Xiuwei Wang. She has always been behind me. She has shared my tiresome years in completing this work. I would also like to thank my son, Sihan, for his understanding and tremendous belief in me. He has played a part in my success. I dedicate this work to them all.

Last but not least, I would like to express my deep appreciation to the faculty, staff and all my fellow graduate students at the department of Computer Science at Concordia University for their assistance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Over the past few decades, the tremendous increase made in the communication network area has been translated into new applications, new devices, and better services, such as WWW, online shopping, and video conferencing, etc. All these developments have further driven the network growth at an exponential rate by any measurement, e.g., the number of users or the amount of traffic. Even though network industries have been working hard on developing new technologies and building new networks, network capacity has always been insufficient. Efficient routing techniques and protocols have been playing an important role in computer network development.

Many new applications require reliable multipoint communications via computer networks. Efficient routing of messages is a key to the performance of network communication. With this increasing amount of communication, traditional simple point-to-point data transmission is no longer appropriate to satisfy the demand of data communication, and multicasting has quickly turned out to be a very significant subject in network communication. It is belived that multicasting will grow substantially in the near future. Multicasting will enable direct marketing, pay TV, pay per view movies, remote surgery, and many other services, besides the well known applications, such as video conference, e-class and online games.

1.2 Multicasting Communication Model

Depending on the number of destinations in the data communication model, there are three types of communication patterns: unicasting, multicasting and broadcasting. Formally, let N be any communication network consisting of n nodes. *Unicasting* is a simple point-to-point (or one-to-one) communication model, which consists of sending a message from one processor to another processor within the network. *Broadcasting* is a one-to-all communication model, which consists of sending a message from one processor in the network to all the remaining processors. *Multicasting* is a one-to-many communication pattern and it is a generalization of one-to-one (unicast) and one-to-all (broadcast) communication patterns. It is a very important interconnection communication pattern that exists in various parallel application algorithms. *Multicasting* requires sending a message from a source to k distinct destinations in network N, for $1 \leq k \leq n$. Obviously, this is the most general type of communication, while unicasting ($k = 1$) and broadcasting ($k = n - 1$) are special cases of multicasting.

Multicasting is a primitive and powerful communication model that allows for the communications to be performed more efficiently than when restricting to the *unicasting* or the *broadcasting* communication model. Multicasting communication is in high demand in the development of many data parallel algorithms, and used in many applications

[1, 5, 8, 12, 13, 14, 15, 19, 20, 21].

The design and performance of multicasting operations depend on several characteristics of the network architecture, including the network topology and switching technique. Regarding network topology, many systems have adopted hypercube topology, and the easily constructed low-dimension meshes and tori are also better suited topologies. The hypercube, mesh and torus are good candidates for general-purpose parallel architecture. They are regarded as graphs and the terms *vertices* or *nodes* are used interchangeably in this thesis for the processors which they represent.

1.3 Switching Techniques

The performance of a parallel computer is largely dependent on the performance of its communication network. A great deal of research has been devoted to developing efficient routing algorithms. The message transmission time greatly depends on the underlying switching technology. Switching is the actual mechanism that removes data from an input channel and places it on an output channel. Four types of switching techniques can be applied to the multicast routing algorithm [16].

- **store-and-forward**: when a packet reaches an intermediate node, the entire packet is stored in a packet buffer. The packet is then forwarded to a selected neighboring node when the neighboring node has an available packet buffer.

  The network latency is $(L/B)D$, where L is the packet length, B is the channel bandwidth, and D is the path length.

- **circuit switching**:

  - circuit establishment phase: a physical circuit is constructed between the source and destination nodes.

  - packet transmission phase: the packet is transmitted along the circuit to the destination.
  - circuit termination phase: the circuit is torn down as the tail of the packet is transmitted.

  The network latency is $(L_c/B)D + L/B$, where L_c is the length of the control packet.

- **virtual cut-through**: a packet is stored at an intermediate node only if the next required channel is busy; otherwise, it is forwarded immediately without buffering.

  The network latency is $(L_h/B)D + L/B$, where L_h is the length of the header field.

- **wormhole routing**: a packet is divided into a number of flits for transmission. The header flit governs the route. As the header advances along the specified route, the remaining flits follow in a pipeline fashion without delay. As soon as a flit has been received by a node, it is sent to the next node in its path without waiting for the remaining flits to arrive. If the header flit encounters a busy channel, all of the flits are blocked until the channel becomes available.

  The network latency is $(L_f/B)D + L/B$, where L_f is the length of each flit.

In circuit switching, virtual cut-through, and wormhole routing, the message transmission time is almost independent of the number of hops between two nodes if the network is contention free. Usually $L_c \ll L$, $L_h \ll L$, and $L_f \ll L$. Hence, their network latency is much smaller than that of the store-and-forward technique. Figure 1.1 compares the communication latency of store-and-forward switching, circuit switching and wormhole routing in a contention free network. In this case, the behavior of virtual cut-through is identical to that of the wormhole routing, so virtual cut-through is not shown explicitly. The figure

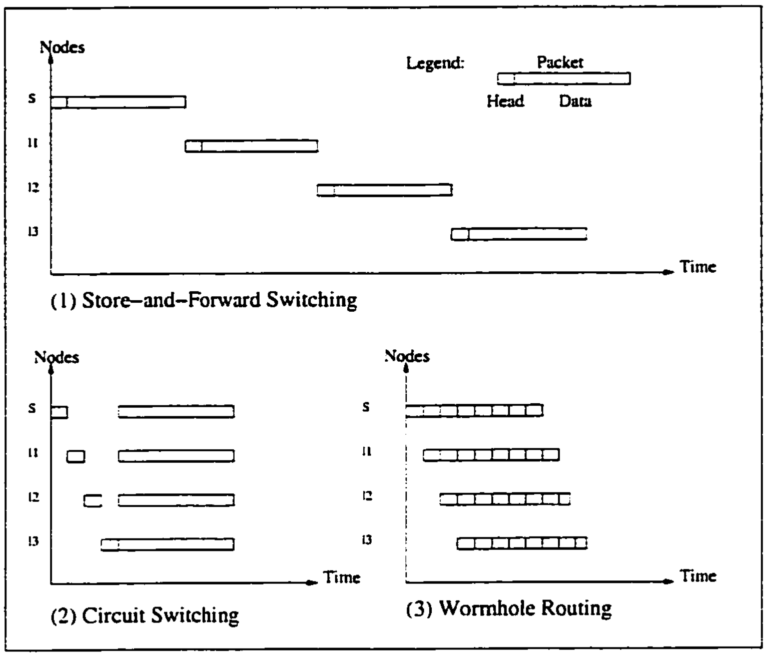


Figure 1.1: Comparison of Switching Techniques

shows the activities of each node over time when a packet is transmitted from a source node S to the destination node l_3 through intermediate nodes l_1 and l_2.

Although message transmission time may be nearly distance-insensitive in a wormhole-routed network, it is still desirable to reduce path lengths whenever possible, since messages that travel on shorter paths use fewer channels. This reduces overall channel loads, which decreases the frequency of channel contention.

Store-and-forward routing can be seen as a fundamental switching technique. In general, store-and-forward routing is a simple technique that works well when the packets are small in comparison with the channel widths. In this thesis, the store-and-forward mechanism will be considered to develop the multicasting algorithms.

1.4 Multicasting in Hypercube

1.4.1 The Properties of Hypercube

Hypercube has become one of the most attractive multiprocessor structures. It contains several parallel processors based on the binary n-cube network. A n-cube parallel processor consists of $N = 2^n$ processors (nodes) addressed by n-bit binary numbers from 0 to $2^n - 1$. Each node has its own memory, and is interconnected with n neighbors.

Hypercube topology has been considered an ideal parallel architecture for its powerful interconnection features. Research on multicast routing in hypercube has received great attention [12, 13, 14, 20, 21]. Now let us formally define the hypercube.

Definition 1.1. *A hypercube (n-cube) consists of $N = 2^n$ nodes (processors) addressed by n-bit binary numbers from 0 to $2^n - 1$. Every node i has n neighboring nodes, where the jth neighbor's address differs in exactly the jth bit position from the node i, for $0 \leq j \leq n-1$, and $0 \leq i \leq N-1$.*

One important property of the n-cube is that it can be constructed recursively from lower dimensional cubes. More precisely, consider two identical $(n-1)$-cubes whose vertices are numbered likewise from 0 to $2^{n-1} - 1$. By joining each vertex of the first $(n-1)$-cube to the same vertex of the second one, one obtains an n-cube. In general, an n-cube can be split into two $(n-1)$-cubes so that the nodes of the two $(n-1)$-cubes are in a one-to-one correspondence. It is clear that there is no cycle of odd length in an n-cube. Concsider a cycle $A_1, A_2, \ldots, A_t$, with $A_1 = A_t$. As the path travels from node A_i to node A_{i+1}, $1 \leq i \leq t-1$, one parity changes. Since $A_1 = A_t$, there must be an even number of changes along the path, e.g., the length of the cycle is necessarily even. Another property of hypercube is that it is a connected graph with diameter n.

Moreover, there are a few simple rules which characterize an n-cube.

Proposition 1.1. *A graph $G = (V, E)$ is an n-cube if and only if*

- *V has 2^n vertices;*
- *every vertex has degree n;*
- *G is connected;*
- *any adjacent nodes A and B are such that the node adjacent to A and those adjacent to B are linked in a one-to-one fashion.*

Let the distance of two processors be the length (number of links) of a shortest path between them. Obviously, the distance between two processors in a hypercube is equal to the Hamming distance of their binary addresses. The Hamming distance between two nodes is defined as the number of corresponding bits differing in their binary addresses. An n-cube can be represented in $n+1$ stages of nodes in incremental Hamming distances from a given source node s, where stage 0 contains s and stage i $(1 \leq i \leq n)$ contains all nodes whose Hamming distance to s is i. In general, assume the source s is at node 0, which is at stage 0. As a result, the nodes at stage 1 have only one 1 in their respective binary addresses, and similarly the nodes at stage i have exactly i 1's in their binary addresses.

Given a source s and k destinations $d_1, d_2, \ldots, d_k$, multicasting requires sending a message from s to d_i for $1 \leq i \leq k$. If we draw the n-cube in stages of incremental Hamming distances from s to d_i, and the route always follow the stage increment order, then the path between s and d_i must be the shortest path.

1.4.2 Heuristic Multicasting Algorithms in the Hypercube Network

It is desirable to develop a routing scheme that minimizes both *time* and *traffic*. It is proven in [14] that the problem of finding such an algorithm in the hypercube network is

NP-complete, but several heuristic multicasting algorithms have been proposed [13, 20, 21].

Lan, Esfahanian, and Ni [13] presented a heuristic algorithm, the Greedy multicast algorithm. The purpose of LEN's algorithm was to generate a low amount of total traffic under the constraints that each destination node can receive the message with a minimum number of hops. The Greedy multicast algorithm is of time complexity $O(nk + n^2)$, where n is the dimension of the hypercube and k is the number of destinations in the multicast.

Sheu and Su [21] developed a heuristic multicast algorithm with time complexity $O(nN)$ in n-dimensional hypercube, where $N = 2^n$. SS's algorithm can be divided into two phases. In phase 1 a message is transmitted from stage 0 to stage $n - 1$, in order to calculate each node's potential weight, which indicates whether or not the node is proper to pass the source message to its children. In phase 2 the multicast paths are found by choosing proper nodes, which have the maximum potential weights, going backwards from stage n to 1. This algorithm guarantees that each destination node can receive the message through the shortest path, which will reduce communication traffic.

Shen, Evans, and You [20] proposed a fault-tolerant multicast algorithm with time complexity $O(nN)$ in the n-dimensional hypercube, where $N = 2^n$. SEY's algorithm uses the lightest node to balance and minimize the traffic, as well as to minimize the number of intermediate (non-destination) nodes. This algorithm minimizes not only the maximum number of hops but also the maximum number of active links connected to any node.

These heuristic multicast algorithms use different strategies and approaches. The *average additional traffic*, defined as the average amount of total traffic minus the number of destination nodes k, can be used to evaluate the performance of these algorithms. The SS's and SEY's algorithm generate similar results, but the LEN's algorithm generates higher average additional traffic than the others (See Figure 1.2).

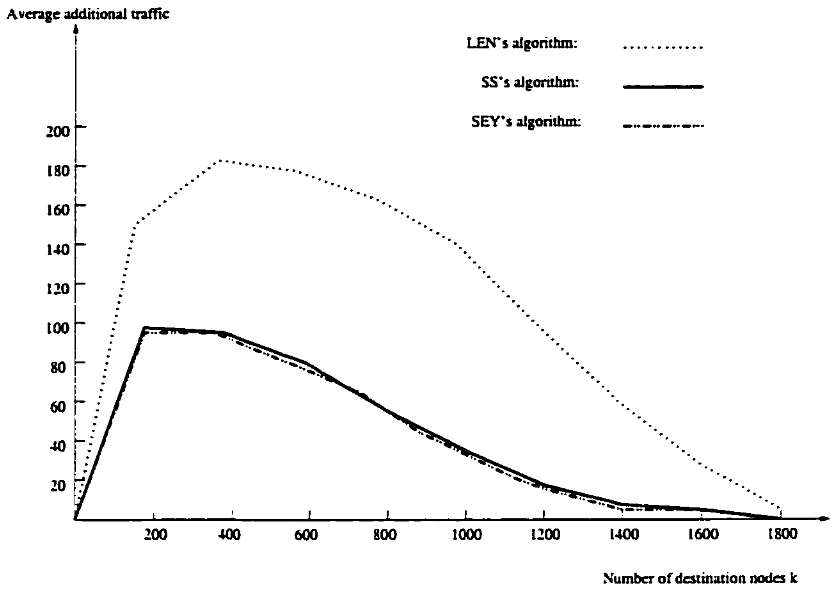


Figure 1.2: Average additional traffic in a 12-cube

1.5 Mesh-connected Networks

Mesh-connected topology is one of the most thoroughly investigated network topologies for parallel processing. Mesh-connected topology is important due to its simple structure and its good performance in practice. These types of topologies, also called k-ary n-cube based networks, have an n-dimensional grid structure with k nodes in each dimension, which include:

- n-dimensional mesh,
- torus (a mesh with wrap-around links),
- hypercube.

These topologies have desirable properties of regularity, balanced behavior, and many alternative paths. (k, n)-meshes (k-ary n-dimensional mesh) and (k, n)-torus (k-ary n-dimensional torus) are common mesh-connected topologies.

Commonly used torus and mesh networks are:

- $(2, n)$-torus, also known as n-dimensional hypercubes (Figure 1.3a when $n = 3$),
- $(k, 2)$-torus, also known as 2-dimensional torus (Figure 1.3b when $k = 5$),
- $(k, 2)$-mesh, also known as 2-dimensional mesh (Figure 1.3c when $k = 5$),
- $(k, 3)$-mesh, also known as 3-dimensional mesh (Figure 1.3d when $k = 5$).

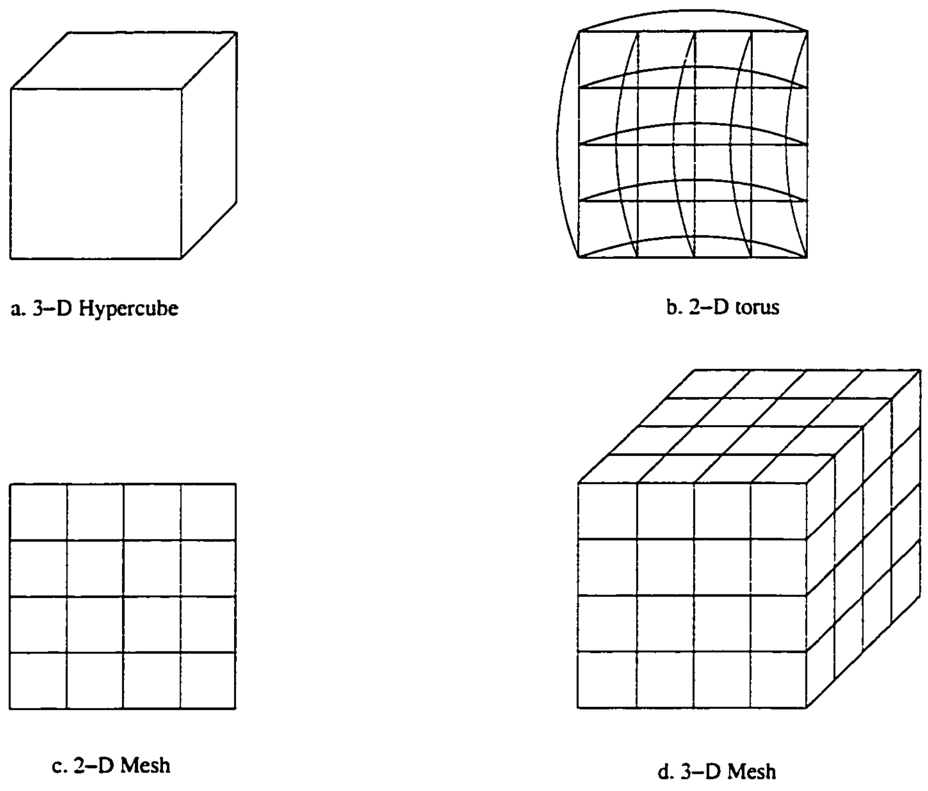


Figure 1.3: Mesh-connected networks

This thesis focuses on 2- and 3-dimensional meshes and torus networks and will give a general idea in the n-dimensional network.

Mesh is another simple topology to implement besides the hypercube topology. Multicasting can be achieved by explicitly joining interested users with mesh architecture. The mesh graph interconnection network has been recognized to be an attractive alternative to the popular hypercube network. One of the most thoroughly investigated interconnection schemes for parallel computation is the $m \times m$ mesh, in which m^2 processing units are connected by a two-dimensional grid of communication links. Its immediate generalizations are n-dimensional $m \times \cdots \times m$ meshes. Despite their large diameters, meshes are of great importance due to their simple structure and efficient layout. The $m \times n$ 2-dimensional mesh, $m \times n \times p$ 3-dimensional mesh, and $K_0 \times K_1 \times \cdots \times K_{n-1}$ n-dimensional meshes are more general cases, and will be considered during this study.

Tori are the variant of meshes in which the nodes on the outside are connected with wraparound links to the corresponding nodes at the other end of the mesh. Tori are node symmetric. All nodes in a torus are identical and no region of the torus is particularly likely to suffer from congestion. Furthermore, the diameters of tori are smaller by a factor of two than those of meshes. Numerous parallel machines, such as the Intel Paragon, Cray T3D, and Cray T3E, have been built with two- and three-dimensional mesh and torus topologies.

For a message delivery in multicasting communication environment, a multicasting algorithm is *optimal* if it minimizes both time and traffic. It is known that finding an optimal multicasting algorithm that minimizes both of *time* and *traffic* is NP-hard in general [14].

It is difficult to minimize both time and traffic, but which one of these two criteria should be minimized first? It depends on the situation of the corresponding network. To illustrate, consider multicasting in a 6×6 mesh in which the message is initially at the source node $s(0,0)$ and $\{(4, 1), (3, 3), (2, 5)\}$ are the destination nodes. If the network traffic is more concerned, the message transmission should follow the path: $(0,0) \Rightarrow (3,0) \Rightarrow (1,1) \Rightarrow (3,3) \Rightarrow (2,5)$ (See figure 1.4 path 1), in which the total number of active links is 10 and the multicasting time is 9. On the other hand, if the transmission time is more important, then the multicasting paths become: $(0,0) \Rightarrow (0,1) \Rightarrow (4,1)$, $(0,0) \Rightarrow (0,3) \Rightarrow (3,3)$, and $(0,0) \Rightarrow (0,5) \Rightarrow (2,5)$ (See figure 1.4 path 2), in which the total multicasting time is

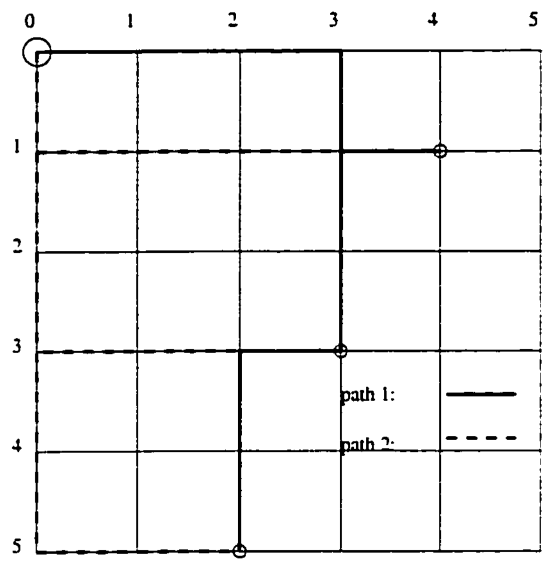


Figure 1.4: A multicasting communication in a 6 × 6 mesh network

reduced to 7, but the total number of active links becomes 14, which is 5 more than the previous one.

These two parameters are not totally independent and achieving a lower bound for one may prevent us from achieving the other. This issue will be discussed in more detail in chapter 2.

1.6 Basic Concept of Gossiping

The gossiping problem is a restricted version of the multimessage multicasting problem. Among other issues, this thesis also presents a result for the gossiping problem in the multicasting communication environment.

Gossiping is a fundamental communication problem: initially each node in a network knows some data, which must be routed so that in the end all nodes have the complete data (this problem is also called *all-to-all broadcast*). *Gossiping* appears as a subroutine in many important problems and is worth studying.

Because of its rich communication pattern, gossiping is a useful benchmark for evaluating the communication capability of an interconnection structure. Gossiping as an embedded operation is needed in many real computations, such as matrix multiplication, LU-factorization, Householder transformation, direct N-body computation, global processor synchronization, and load balancing. Gossiping problems have been studied under many different objective functions and communication models. Our communication model allows each processor to multicast one message to any subset of its adjacent processors, but no processor may receive more than one message at a time. Our objective is to determine when each of these messages is transmitted so that the communication can be carried out in the minimum amount of time. In a communication network that contains n processors, initially each processor holds a message, and requires the remaining $n-1$ messages. Under our communication model, every processor needs to receive $n-1$ messages and no processor may receive two or more messages simultaneously, which implies that $n-1$ is a lower bound on the total communication time of gossiping.

If the network contains a Hamiltonian circuit, to perform the gossiping communication, each processor sends to its clockwise neighbor the message it holds in the first step, and then in the remaining iterations every processor transmits to its clockwise neighbor the message it just received from its counter-clockwise neighbor. The total communication time is $n-1$, which is the optimal solution. Unfortunately, to find a Hamiltonian circuit in a graph is an NP-complete problem in general. However, it is not necessary for a network to have a Hamiltonian circuit in order to solve the gossiping problem in $n-1$ steps. The gossiping problem can be finished in $n-1$ time units for other networks as well. The Petersen graph is one such example [7]. At the end of this thesis we will present a graph on n nodes (called

H) with $n-1$ gossiping time. Then we present a polynomial algorithm to recognize whether a given network has a spanning subgraph H.

1.7 Thesis Organization

The organization of the rest of the thesis is as follows. Chapter 2 proposes the VH and $DIST$ multicasting algorithms in 2-dimensional mesh and torus networks, then expands the algorithms to 3-dimensional mesh and torus networks, and finally explores a general idea for multi-dimensional mesh and torus networks. The chapter also discusses the relationship between the VH and $DIST$ algorithms.

Chapter 3 implements the algorithms for 2- and 3-dimensional mesh and torus networks for VH algorithm and $DIST$ Algorithm. In addition, some comparative results are given. The VH Algorithm uses less multicasting time, but the $DIST$ Algorithm has much better average traffic performance.

Chapter 4 analyzes the design issue and explains the complexities of these algorithms.

Chapter 5 designs an algorithm which recognizes a subgraph in an arbitrary graph, and then proves such a subgraph can perform the gossiping communication in the minimal possible time $n-1$.

Chapter 6 concludes with a summary of this thesis and highlights some future extensions.

Chapter 2

Multicasting Algorithms in Mesh and Torus Networks

2.1 Preliminaries

The main problem in multicasting communication is that of determining which paths should be used to deliver a message from the source to all its destinations. Since there are many potential paths, different routes can be found, depending on the criteria employed.

In this section, some notation and definitions, and a general model for multicasting communication are presented. The graph theory terminology and notation will be followed. Terms not defined here can be found in Harary's book [9]. Let $G(V, E)$ be a *graph* with the node set $V(G) = V$ and edge set $E(G) = E$. If an edge $e = (u, v) \in E$, then nodes u and v are said to be *adjacent*. The term *edge* and *link* are used interchangeably.

A path is an alternating sequence of nodes and edges, beginning and ending with nodes, in which all nodes are distinct. A simple path p from node u_0 to node u_k is represented by an ordered sequence of nodes $(u_0, u_1, \ldots, u_k)$. The length of path p is measured by the number of edges contained in the path. For all algorithms discussed in this thesis, no link will be traversed by the same message more than once. Therefore the value of total traffic is equal to the number of links involved in the multicast. Thus the above path has length k.

When considering communication issues at the system's level, the main problem is that of determining which paths should be used to deliver a message from a node (called the *source node*) to some *destination nodes*. This path selection process is commonly refered to as *routing*. *Time* and *traffic* are the major routing design parameters considered when adopting a multicasting communication scheme. *Time* is measured in the actual time steps needed to send a message from the source to a destination. *Traffic* is quantified by the number of messages traversed in the communication links that are used to deliver the source message to its destinations. These two parameters are not totally independent of each other. The VH algorithm delivers the message in the minimal time cost, whereas the $DIST$ algorithm can reduce the traffic for a price of increasing the multicasting time.

Mesh-connected network topologies are well known network architectures. This thesis focuses on mesh and torus networks, as they have similarities [3]. Mesh and torus interconnection networks are able to support many scientific and image processing applications efficiently. The following introduces some definitions and notations related to the torus and mesh networks.

2.1.1 Mesh Network

Definition 2.1. *Formally, let us define an n-dimensional mesh* $(K_0 \times K_1 \times \cdots \times K_{n-1})$, *where* $K_i \geq 2$ *for* $0 \leq i \leq n-1$. *The mesh network contains* $N = \prod_{i=0}^{n-1} K_i$ *nodes. Each node in the mesh has a unique label of the form* $(x_0, x_1, \ldots, x_{n-1})$, *where* $0 \leq x_i \leq K_i - 1$

for all i, $0 \leq i \leq n-1$. In dimension i, $0 \leq i \leq n-1$, the connectivity for node $X(x_0, \ldots, x_i, \ldots, x_{n-1})$ is:

$$X \longrightarrow \begin{cases} (x_0, \ldots, x_i + 1, \ldots, x_{n-1}), & \textit{if } x_i < K_i - 1, \\ (x_0, \ldots, x_i - 1, \ldots, x_{n-1}), & \textit{if } x_i > 0. \end{cases}$$

In the mesh network, the distance between node $(x_0, \ldots, x_i, \ldots, x_{n-1})$ and node $(y_0, \ldots, y_i, \ldots, y_{n-1})$ is $\sum_{i=0}^{n-1} |y_i - x_i|$. The maximum degree of the node in the n-dimensional mesh is $2n$, and the diameter of the n-dimensional mesh is $\sum_{i=0}^{n-1} K_i - n$.

Consider an $m \times n$ 2-dimensional mesh (2DM). Each node is identified by coordinate (x, y), where $0 \leq x \leq m-1$ and $0 \leq y \leq n-1$. The total number of nodes in 2DM is $N = m \times n$. The connectivity for node (x, y) is:

$$(x, y) \longrightarrow \begin{cases} (x+1, y), & \text{if } x < m-1, \\ (x-1, y), & \text{if } x > 0, \\ (x, y+1), & \text{if } y < n-1, \\ (x, y-1), & \text{if } y > 0. \end{cases}$$

In 2DM, every node has at least 2 neighbors and at most 4. Thus, the node of 2-D mesh has degree at least 2 and at most 4, and the diameter of 2DM is $m + n - 2$ (see Figure 2.1 (a)).

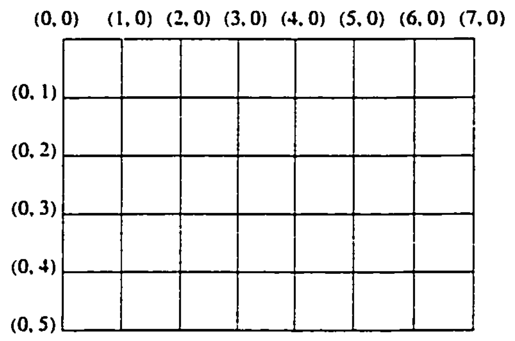
The 3-dimensional mesh (3DM) is another commonly used mesh structure. A 3-D mesh network has the form of $(m \times n \times p)$, which contains $N = m \times n \times p$ nodes. The node of 3-D mesh has maximum degree 6, and the diameter of 3DM is $m + n + p - 3$. Each node is identified by coordinate (x, y, z), where $0 \leq x \leq m-1$, $0 \leq y \leq n-1$, and $0 \leq z \leq p-1$.

The connectivity for node (x, y, z) is:

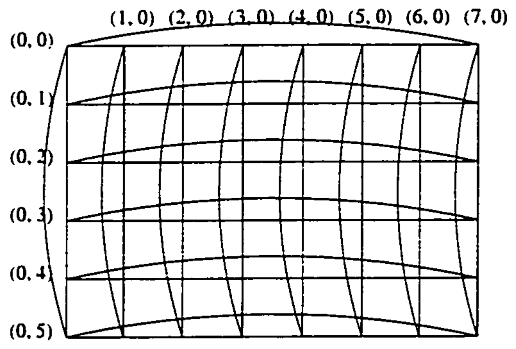
$$(x, y, z) \longrightarrow \begin{cases} (x+1, y, z), & \text{if } x < m-1, \\ (x-1, y, z), & \text{if } x > 0, \\ (x, y+1, z), & \text{if } y < n-1, \\ (x, y-1, z), & \text{if } y > 0, \\ (x, y, z+1), & \text{if } z < p-1, \\ (x, y, z-1), & \text{if } z > 0. \end{cases}$$

2.1.2 Torus Network

The torus network is another commonly used mesh-connected network. It has some advantages over the mesh graph. In general, a torus network is a mesh with wrap-around connection in every dimension. Torus networks, which can support a larger number of users than common linear topology networks, have been proposed for metropolitan area network (MAN) architectures. The torus network has approximately half the diameter of the mesh network. It can play an important role in the next generation of parallel computers.



(a) 8 X 6 mesh graph



(b) 8 X 6 torus graph

Figure 2.1: 2-dimensional mesh and torus networks

Definition 2.2. *The torus is identical to the mesh, except for the connectivity. Let us define an n-dimensional torus $(K_0 \times K_1 \times \cdots \times K_{n-1})$, where $K_i \geq 2$ for $0 \leq i \leq n-1$. The torus network contains $N = \prod_{i=0}^{n-1} K_i$ nodes. Each node in the torus has a unique label of the form $(x_0, x_1, \ldots, x_{n-1})$, where $0 \leq x_i \leq K_i - 1$ for all i, $0 \leq i \leq n-1$. In dimension i, $0 \leq i \leq n-1$, the connectivity for node $X(x_0, \ldots, x_i, \ldots, x_{n-1})$ is:*

$$X \longrightarrow \begin{cases} (x_0, \ldots, (x_i + 1) \, mod \, K_i, \ldots, x_{n-1}), \\ (x_0, \ldots, (x_i - 1) \, mod \, K_i, \ldots, x_{n-1}). \end{cases}$$

In a torus network, the distance between node $(x_0, \ldots, x_i, \ldots, x_{n-1})$ and node $(y_0, \ldots, y_i, \ldots, y_{n-1})$ is $\sum_{i=0}^{n-1} min(|y_i - x_i|, K_i - |y_i - x_i|)$. The degree of a node in the n-dimensional torus is $2n$ and its diameter is $\sum_{i=0}^{n-1} \lfloor \frac{K_i}{2} \rfloor$.

A 2-D torus networks (2DT) has the form of $(m \times n)$, which has $N = m \times n$ nodes. Each node is identified by coordinate (x, y), where $0 \leq x \leq m-1$ and $0 \leq y \leq n-1$. Every node has exactly four neighbors, so each node of a 2-D torus has degree 4, and the diameter of 2DT is $\lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor$. The connectivity for node (x, y) is:

$$(x, y) \longrightarrow \begin{cases} ((x \pm 1) \, mod \, m, y), \\ (x, (y \pm 1) \, mod \, n). \end{cases}$$

In an $m \times n \times p$ 3-D torus network (3DT), each node is identified by coordinate (x, y, z), where $0 \leq x \leq m-1$, $0 \leq y \leq n-1$, and $0 \leq z \leq p-1$. Every node has exactly six neighbors in 3-D torus networks, so the node of 3-D torus has a degree of 6, and the diameter of 3DT is $\lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + \lfloor \frac{p}{2} \rfloor$. The connectivity for node (x, y, z) is:

$$(x, y, z) \longrightarrow \begin{cases} ((x \pm 1) \, mod \, m, y, z), \\ (x, (y \pm 1) \, mod \, n, z), \\ (x, y, (z \pm 1) \, mod \, p). \end{cases}$$

In n-dimensional torus networks, the link connecting nodes $(x_0, \ldots, x_{i-1}, K_i - 1, x_{i+1}, \ldots, x_{n-1})$ and $(x_0, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_{n-1})$ are called wraparound links. All of the links are bidirectional. A transfer is in the *positive direction* if a packet transfers from a node $(x_0, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_{n-1})$ to a node $(x_0, \ldots, x_{i-1}, x_i + 1 \, mod \, K_i, x_{i+1}, \ldots, x_{n-1})$, and transfers in the opposite direction will be refered to as the *negative direction.*

The mesh and torus networks $(K_0 \times K_1 \times \cdots \times K_{n-1})$ have exactly the same number of nodes $N = \prod_{i=0}^{n-1} K_i$. However, the torus network has approximately half the diameter of the mesh network.

2.1.3 Multicasting Communication Model

Before presenting the multicasting algorithms, let us formally define the multicasting communication model used in mesh and torus networks. Let N be any mesh or torus communication network (or graph). Initially the message holds in the originator and a set of destination nodes need the message. The multicasting communication model satisfy the following restrictions:

- During each time unit one processor may transmit the message to only one of its neighboring nodes.
- During each time unit each processor may receive at most one message.
- During each time unit a message can be transmitted over different links simultaneously.

The communication process ends when the set of destination nodes has received the desired message.

2.2 2-Dimensional Mesh Network

2.2.1 *VH* Algorithm in 2-Dimensional Mesh Network

It is easy to see that the multicasting time is equal to or less than the total nmber of links for any multicasting algorithm.

As discussed earlier, it is desirable to develop a multicasting communication algorithm that minimizes both time and traffic, although this is known to be NP-hard [14]. The first approach, the dimension-order algorithm (VH algorithm), will minimize the time first. In the VH algorithm, a message is transmitted first in the highest dimension in which the source and destination nodes differ. Routing then proceeds on each required dimension, in descending order of dimension, until the routing path reaches the source. The routing paths in the VH algorithm always follow the shortest paths, which guarantees the minimal multicasting time.

The 2-D mesh (2DM) is a basic structure in a mesh-connected network. Let us define a 2-D mesh network ($m \times n$), given a source $s(0,0)$ and a set of destination nodes $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$, where $0 \leq x_i \leq m-1$ and $0 \leq y_i \leq n-1$. Multicasting requires sending a message from s to all (x_i, y_i).

The Vertical-Horizontal oriented algorithm (VH algorithm) uses a backward dimensional approach, which starts from the destination node, and finds a backward path until reaching the source node $s(0,0)$. The routing follows the vertical dimension (y dimension) first, then turns to the horizontal dimension (x dimension). Routing in the y dimension is always complete before routing in the x dimension starts.

There is an alternative routing approach, which follows the x dimension first, and then turns to the y dimension until reaching the source $s(0,0)$. The number of links generated by these two different routings may vary. For example, in a 8×6 2DM, a message is sent from node $(0,0)$ to nodes $\{(6,2),(3,4)\}$. If the Vertical-Horizontal oriented routing is used, the multicasting time is 8, and the number of links is 12. If the Horizontal-Vertical oriented routing is applied, the multicasting time is also 8, but the number of links increases to 13. It is better to compare the results of these two routing, and choose the one which generates less traffic. Without loss of generality, we only consider the Vertical-Horizontal oriented approach in this thesis.

Definition 2.3. *In a 2DM $(m \times n)$, consider the source node $s(0,0)$, and a set of destination nodes $\{(x_1,y_1),(x_2,y_2),\ldots,(x_i,y_i)\}$, where $0 \le x_i \le m-1$ and $0 \le y_i \le n-1$:*

$x_{max} = max\{x_1,\ldots,x_i\}$

$y_{max} = max\{y_1,\ldots,y_i\}$

$D_{max} = max\{D_1,\ldots,D_i\}$

where D_j is the distance between the source and destination (x_j,y_j), for $1 \le j \le i$.

Algorithm 2.1 (The VH_{2DM} Algorithm). *In a 2D mesh network $(m \times n)$, given a source $s(0,0)$ and a set of destinations $\{(x_1,y_1),(x_2,y_2),\ldots,(x_i,y_i)\}$, where $0 \le x_i \le m-1$ and $0 \le y_i \le n-1$. Multicasting requires a message to be sent from s to all (x_i,y_i)s.*

1. *Assign distances (D_j) for all destination nodes $(D_j = x_j + y_j, for\, 1 \le j \le i)$.*

2. *Construct the multicasting tree.*

   - *Start from the destination with value x_{max}, and then expand the path backward.*
   - *Follow the vertical dimension (y dimension) until the node $(x_j,0)$.*
   - *Continue along the horizontal dimension (x dimension) until the source node $s(0,0)$ is reached.*
   - *Connect the destination (x_j,y_j) to node $(x_j,0)$ for all destination nodes.*

3. *If more than one node has the same x value, choose the one that has the biggest y value.*

Under the multicasting communication model, the minimal multicasting time is equal to the value of maximum distance D_{max} or $D_{max} + 1$.

Proposition 2.1. *In the network communication, consider a source $s(0,0)$ and a set of destination nodes $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$, where $0 \leq x_i \leq m-1$ and $0 \leq y_i \leq n-1$. The minimal multicasting time satisfies:*

$$TIME = \begin{cases} D_{max}, & \textit{if only one node has value } D_{max}; \\ D_{max} + 1, & \textit{if more than one node has value } D_{max}. \end{cases}$$

Proof. To reach every destination node in a graph, the shortest path must be followed. In general, the multicasting time must be equal to or greater than the maximum distance D_{max} for any algorithm. In any given algorithm, to send a message from node $(0,0)$ to the node with distance D_{max}, at least D_{max} time units is needed. If there is more than one node having the value D_{max}, at least one more time unit is needed to reach all the destinations. Thus multicasting time must be equal to or greater than D_{max} for any multicasting algorithm. □

Proposition 2.2. *The VH_{2DM} Algorithm always generates the minimal multicasting time.*

$$TIME_{VH_{2DM}} = \begin{cases} D_{max}, & \textit{if only one node has value } D_{max}; \\ D_{max} + 1, & \textit{if more than one node has value } D_{max}. \end{cases} \tag{2.1}$$

Proof. In the VH_{2DM} Algorithm, every destination node follows the shortest path. Therefore, the VH_{2DM} algorithm generates the minimum possible multicasting time D_{max} when only one node has value D_{max} or $D_{max} + 1$ when more than one node has value D_{max}. □

Assume the set of destination nodes are $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$, where $x_1 \leq x_2 \leq \cdots \leq x_i$, and if nodes $\{(x_j, y_j), \ldots, (x_k, y_k)\}$ have the same x value assume $y_j \leq y_{j+1} \leq \cdots \leq y_k$ for $1 \leq j, k \leq i$. If only one destination has the maximum distance value, the multicasting time is equal to the value of maximum distance D_{max}. Otherwise, if more than one destination has the maximum distance value, the algorithm needs one additional time unit to finish multicasting (see the example in Figure 2.3, 2.4)

The total number of links also depends on the pattern of destinations. If the x value of each node is unique, the number of links will count every y value. If some nodes have the same x value, then only the maximum y value among these nodes is counted.

Example 2.1. *In a 2DM network, consider the source $s(0,0)$ and a set of destinations $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6)\}$. Nodes (x_1, y_1) and (x_2, y_2) have the same x value, $x_1 = x_2$. When counting the total traffic, y_1 is not counted in the total. Nodes $(x_4, y_4), (x_4, y_4), (x_6, y_6)$ have the same x value, $x_4 = x_5 = x_6$. For the same reason, y_4 and y_5 are not counted either. So the value $LINK_{VH} = x_6 + \sum(y_2 + y_3 + y_6)$ (See Figure 2.2).*

The total number of links satisfies $LINK_{VH_{2DM}} \leq x_i + \sum_{j=1}^{i} y_j$. To calculate the exact sum of total y values, the following procedure can be used.

Procedure 2.1. *The sum of y values.*

```
$\sum_{2DM} = 0$;
do
  take node $(x_j, y_j)$
  if $x_j$ is unique,
      then include $y_j$ in the sum, $\sum_{2DM} = \sum_{2DM} + y_j$.
```

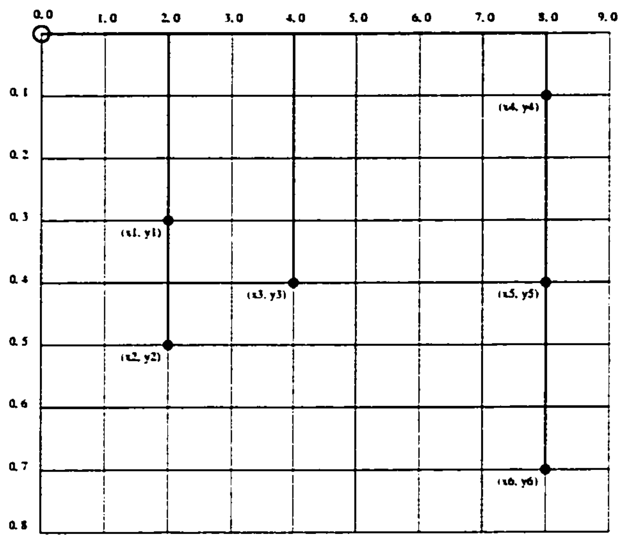


Figure 2.2: Multicasting in the VH Algorithm

```
    else if $x_j = x_{j+1} = \cdots = x_p$
        then include only $y_p$ in the sum, $\sum_{2DM} = \sum_{2DM} + y_p$.
        (do not include the value of $y_j, y_{j+1}, \ldots, y_{p-1}$)
enddo
output $\sum_{2DM}$.
```

Thus the toal number of links can be described as

$$LINK_{VH_{2DM}} = x_i + \sum . \tag{2.2}$$

where $x_i = x_{max}$ and $\sum$ is the output of Procedure 2.1.

The minimum value of $LINK_{VH_{2DM}}$ will be obtained in the case where all the x values are the same, $x_1 = x_2 = \cdots = x_i$, then $LINK_{VH_{2DM}} = x_i + y_i$. The maximum value of $LINK_{VH_{2DM}}$ will be obtained in the case where all x_i's are mutually different, say $x_1 \neq x_2 \neq \cdots \neq x_i$, then $LINK_{VH_{2DM}} = x_i + \sum_{j=1}^{i} y_j$. This means that the number of links satisfies: $x_i + y_i \leq LINK_{VH_{2DM}} \leq x_i + \sum_{j=1}^{i} y_j$.

To illustrate how the VH_{2DM} algorithm works, here are two examples:

Example 2.2. *In a 2DM network* (10×9), *the message is sent from source node* $s(0,0)$ *to a set of destination nodes* $\{(2,2),(4,3),(5,4),(6,6),(7,6)\}$. *By using the* VH_{2DM} *Algorithm, every destination follows the vertical dimension first, and then turns to the horizontal dimension. The actual routing follows,*

$(0,0) \Rightarrow (7,0) \Rightarrow (7,6)$,

$(6,0) \Rightarrow (6,6)$,

$(5,0) \Rightarrow (5,4)$,

$(4,0) \Rightarrow (4,3)$,

$(2,0) \Rightarrow (2,2)$.

The maximum x *value is* $x_{max} = 7$, *and* $D_{max} = 7 + 6 = 13$. *Since only node* $(7,6)$ *has the maximum distance, the multicasting time* $TIME_{VH_{2DM}} = D_{max} = 13$, *and the total number of links* $LINK_{VH_{2DM}} = x_i + \sum_{j=1}^{i} y_j = 28$. *The resulting routing scheme is shown in Figure 2.3.*

In the next example, more than one node has the maximum distance value.

Example 2.3. *Consider a 2DM network* (10×9), *in which a message is sent from source node* $s(0,0)$ *to a set of destination nodes* $\{(2,2),(6,7),(7,3),(7,6),(9,4)\}$. *Applying the* VH_{2DM} *Algorithm, every destination follows the vertical dimension first, then turns to the horizontal dimension. The actual routing follows,*

$(0,0) \Rightarrow (9,0) \Rightarrow (9,4)$,

$(7,0) \Rightarrow (7,3) \Rightarrow (7,6)$,

$(6,0) \Rightarrow (6,7)$,

$(2,0) \Rightarrow (2,2)$.

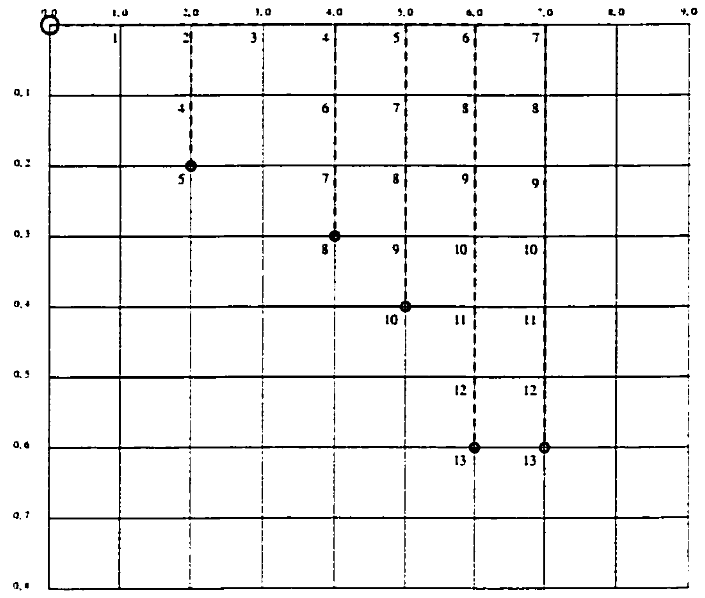


Figure 2.3: Routing scheme in the VH_{2DM} algorithm

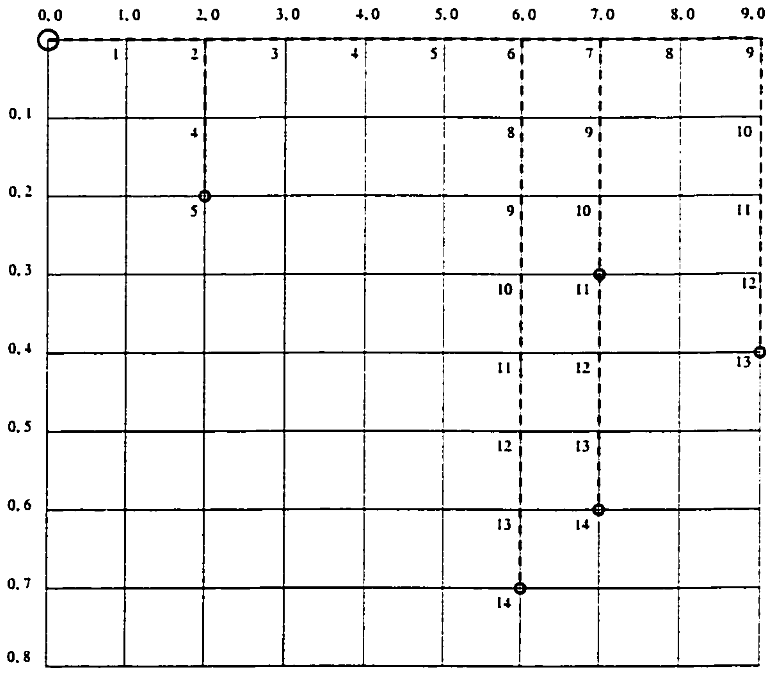


Figure 2.4: Multicast routing in the VH_{2DM} algorithm

The maximum distance is $D_{max} = 6 + 7 = 13$, and nodes $(6,7)$, $(7,6)$, and $(9,4)$ have the distance value of 13. The multicasting time $TIME_{VH_{2DM}} = D_{max} + 1 = 14$. Nodes $(7,3)$ and $(7,6)$ have the same x value of 7, the y value of node $(7,3)$ doest not count in the total number of links. The output of procedure 2.1 is $\sum = 19$ and the maximum x value is $x_i = 9$, so that the number of links $LINK_{VH_{2DM}} = x_i + \sum = 28$. The resulting routing is shown in Figure 2.4.

Let us see another approach to reduce the total traffic in the next section.

2.2.2 *DIST* Algorithm in the 2-Dimensional Mesh Network

It is desirable to minimize the multicasting time, however this does not always guarantee the best performance. In fact, for certain problems, non-minmal routing algorithms can utilize more of the available network bandwidth and cause less communication congestion.

The *DIST* Algorithm uses a different strategy to multicast the message. In the *VH* Algorithm, each destination finds the route independently, while in the *DIST* Algorithm the route for each destination depends on the existing multicasting tree. Every destination node finds the shortest path to the existing tree. This algorithm generates less traffic, but may result in a greater multicasting time as compared with the *VH* Algorithm.

Algorithm 2.2 (The $DIST_{2DM}$ Algorithm). *In a 2DM network $(m \times n)$, given a source s(0, 0) and a set of destinations $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$, where $0 \le x_i \le m - 1$ and $0 \le y_i \le n - 1$. Multicasting requires a message to be sent from s to all (x_i, y_i)s.*

1. *Assign distances (D_j) for all destinations $(D_j = x_j + y_j, where\, 1 \le j \le i)$.*

2. *Build the routes to connect all the destinations to the source:*

    - *Start from the destination with the minimal distance.*

- *Find the shortest path to the existing multicasting tree.*

- *Repeat the process in ascending order of distance until every destination node is included in the multicasting tree.*

3. *If more than one node has the same distance, alternatively take (x_j, y_j) for which x_j is minimum possible value and take (x_k, y_k) for which x_k is maximum possible value.*

The total number of links varies and depends on the pattern of destinations, but the upper bound is given by the expression below. 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-1}, y_{j-1}).

$LINK_{DIST_{2DM}} \leq (x_1 + y_1) + (|x_2 - x_1| + |y_2 - y_1|) + \cdots + (|x_j - x_{j-1}| + |y_j - y_{j-1}|)$.

This means that

$$LINK_{DIST_{2DM}} \leq (x_1 + y_1) + \sum_{j=2}^{i}(|x_j - x_{j-1}| + |y_j - y_{j-1}|) \tag{2.3}$$

The minimum value of $LINK_{DIST_{2DM}}$ will be obtained in the case where all nodes have the same x value, $x_1 = x_2 = \cdots = x_i$, then $LINK_{DIST_{2DM}} = x_1 + y_1 + \sum_{j=2}^{i}(y_j - y_{j-1}) = x_i + y_i$, or in the case where all nodes have the same y value, $y_1 = y_2 = \cdots = y_i$, then $LINK_{DIST_{2DM}} = x_1 + y_1 + \sum_{j=2}^{i}(x_j - x_{j-1}) = x_i + y_i$ (See example in Figure 2.5). The maximum value of $LINK_{DIST_{2DM}}$ can reach $LINK_{DIST_{2DM}} = x_i + \sum_{j=1}^{i} y_j$ (See example in Figure 2.6).

Here, the total time cost seems greater than that of the VH_{2DM} algorithm. The lower bound of multicasting time is D_{max}. In the worst case, the value of $TIME_{DIST}$ is close to that of $3D_{min}$, which is much greater than that of the VH_{2DM} algorithm. For example, consider a (10×9) 2DM, where a message is sent from source $(0,0)$ to a set of destinations $\{(9,2), (8,3), (6,5), (3,8)\}$. All the four destination nodes have the same maximum distance

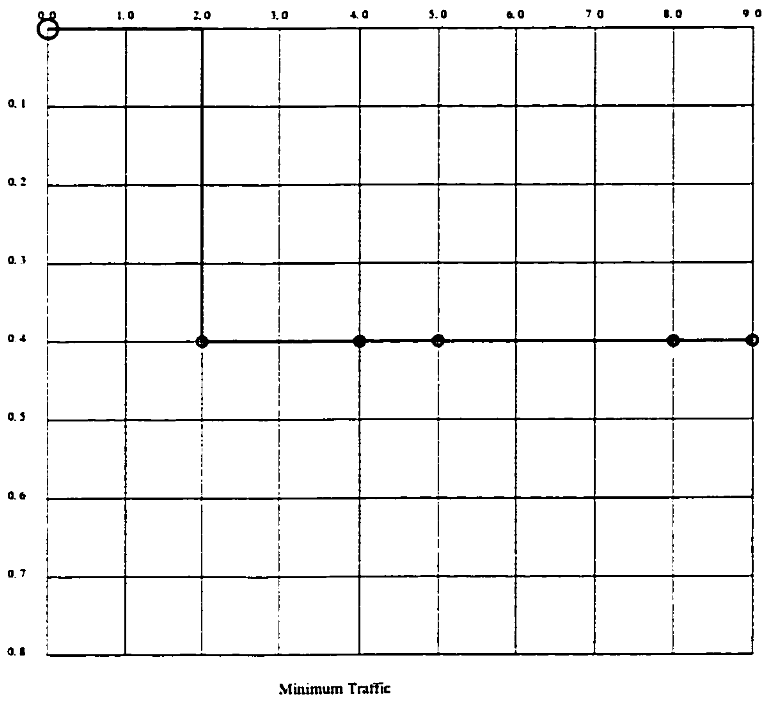


Figure 2.5: The minimum traffic in the $DIST$ Algorithm

value $D_{max} = 11$. If the chosen nodes are in the order $(9,2),(8,3),(6,5),(3,8)$, the resulting routing is $(0,0) \Rightarrow (9,0) \Rightarrow (9,2) \Rightarrow (8,2) \Rightarrow (8,3) \Rightarrow (6,3) \Rightarrow (6,5) \Rightarrow (3,5) \Rightarrow (3,8)$. Multicasting costs 23 time units, which is close to the value of $3D_{max}$. If the destination nodes are chosen in a different sequence, such as $(3,8),(9,2),(6,5),(8,3)$, the resulting routing becomes $(0,0) \Rightarrow (3,0) \Rightarrow (3,8)$, $(3,2) \Rightarrow (9,2)$, $(8,2) \Rightarrow (8,3)$, and $(6,2) \Rightarrow (6,5)$. The multicasting time is 12, which is much less than the previous one (see Figure 2.7).

Therefore, the multicasting time in the $DIST_{2DM}$ satisfies,

$$D_{max} \leq TIME_{DIST_{2DM}} < min\{2m+n-3, 2n+m-3, 3D_{max}\} \qquad (2.4)$$

To illustrate the $DIST_{2DM}$ algorithm, the same examples of the VH_{2DM} algorithm are used to compare the relationship between these algorithms.

Example 2.4. *In a 2DM network* (10×9), *the message is sent from source* $s(0,0)$ *to a set*

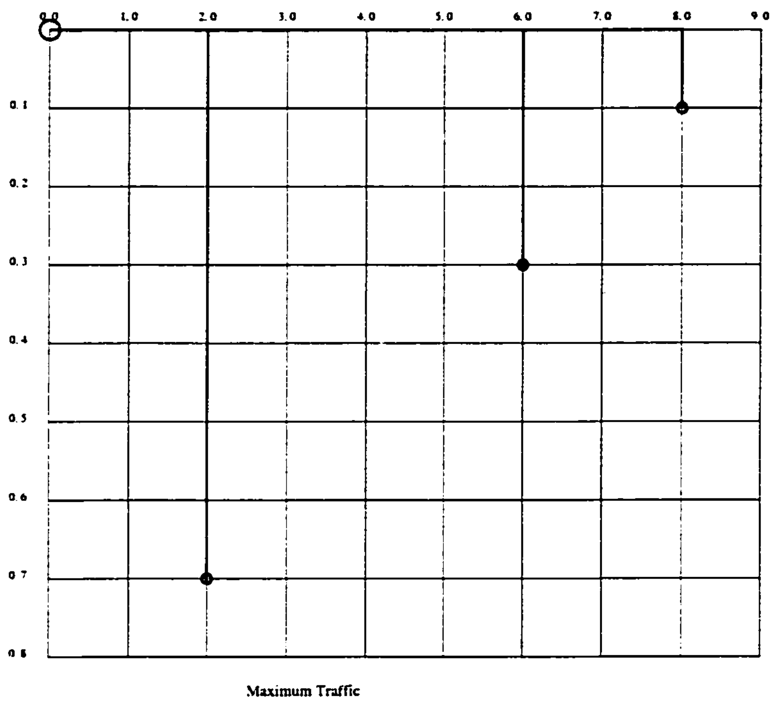


Figure 2.6: The maximum traffic in the $DIST$ Algorithm

of destinations $\{(2,2),(4,3),(5,4),(6,6),(7,6)\}$. *Using the* $DIST_{2DM}$ *Algorithm,* $D_{max} = 7+6 = 13$. *Node* $(2,2)$ *has the minimum distance value of 4, find a path between* $(0,0)$ *and* $(2,2)$ *first, then find the shortest path for node (4, 3) to the existing multicasting tree, which is* $(2,2) \Rightarrow (4,2) \Rightarrow (4,3)$. *Apply for node* $(5,4)$ *next, in which* $(4,3) \Rightarrow (5,3) \Rightarrow (5,4)$ *is the routing path. Repeat this procedure for node (6, 6) and (7, 6). The resulting routing is shown in Figure 2.8.*

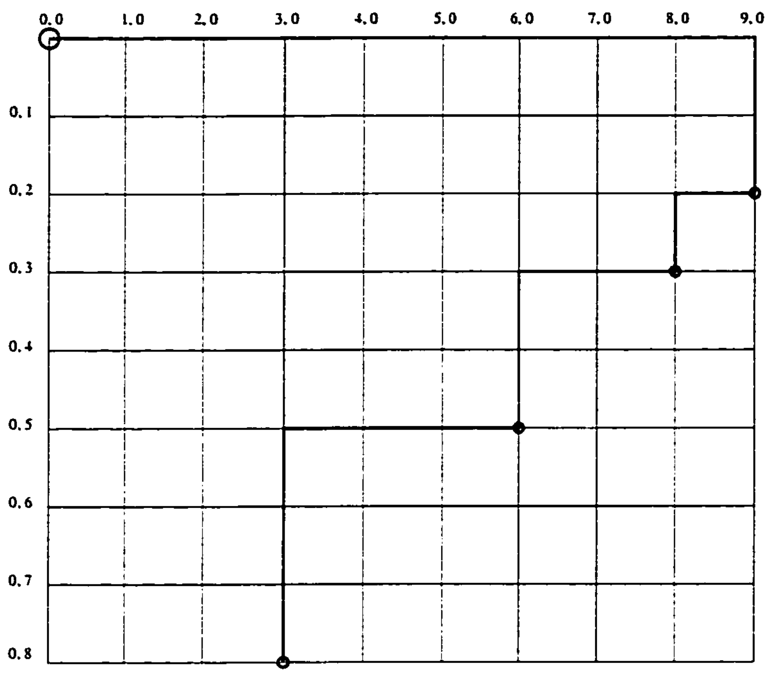
The multicasting time $TIME_{DIST_{2DM}} = D_{max} = 13$.

The number of links $LINK_{DIST_{2DM}} = 13 < x_i + \sum_{j=1}^{i} y_j$.

Comparing the parameters time and traffic with those from example 2.2, the relationship between these algorithms in these particular cases can be concluded as:

$TIME_{VH_{2DM}} = TIME_{DIST_{2DM}}$,

$LINK_{VH_{2DM}} > LINK_{DIST_{2DM}}$.



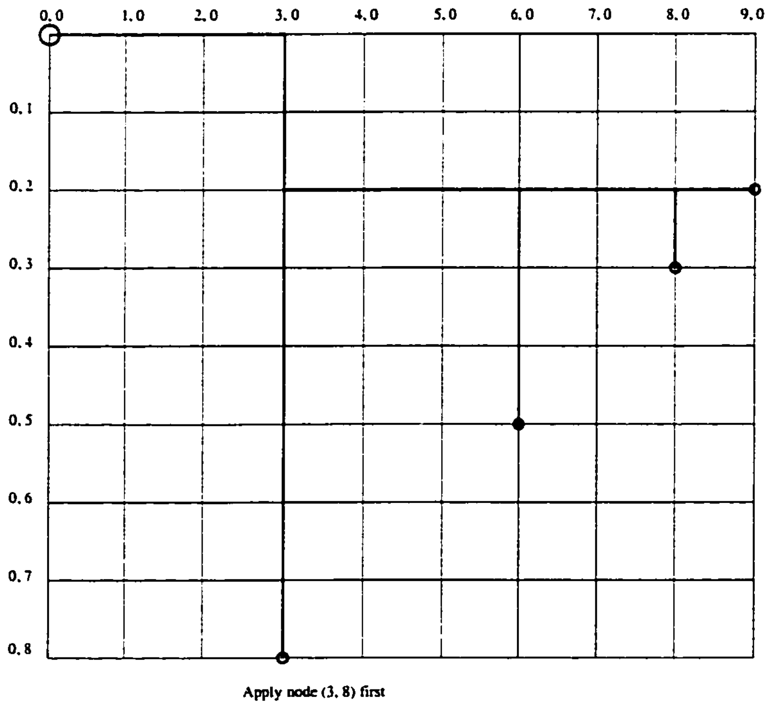


Figure 2.7: Worst case multicasting in the $DIST_{2DM}$ algorithm

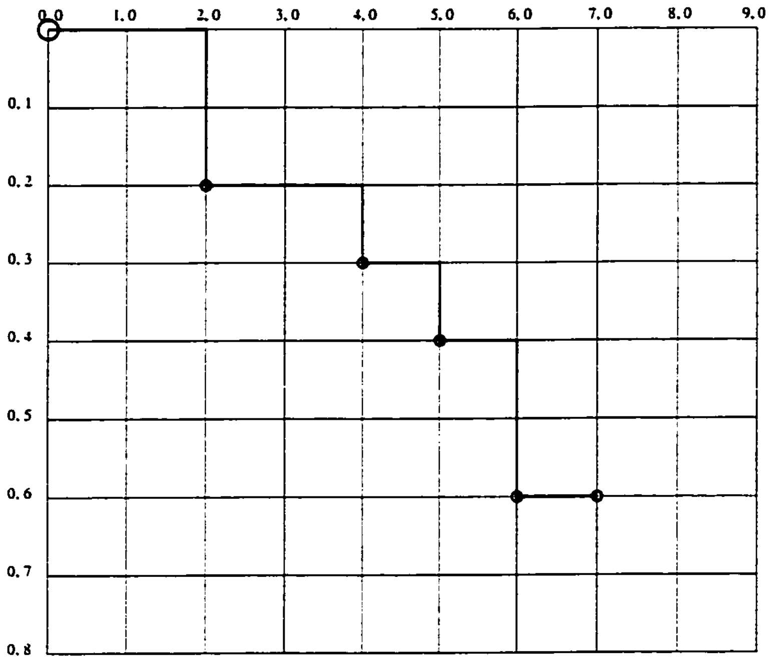


Figure 2.8: Multicast routing scheme in the $DIST_{2DM}$ algorithm

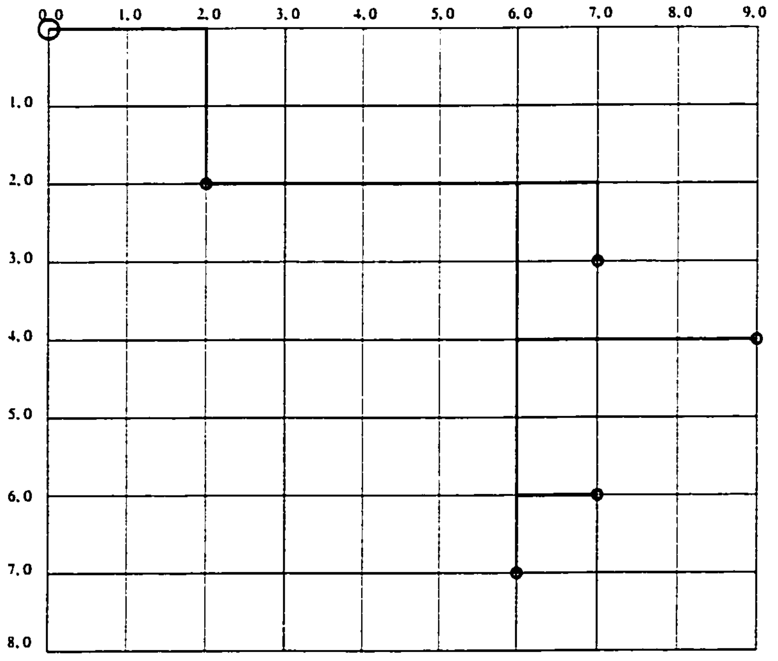


Figure 2.9: Routing scheme in the $DIST_{2DM}$ algorithm

Example 2.5. *In a 2DM network* (10×9), *the message is sent from node* $s(0,0)$ *to a set of destination nodes* $\{(2,2),(6,7),(7,3),(7,6),(9,4)\}$. *By using the* $DIST_{2DM}$ *Algorithm,* $D_{max} = 6+7 = 13$, *and three nodes* $\{(6,7),(7,6),(9,4)\}$ *have the value of* D_{max}. *The first step is the same as in example 2.3: find a path for node* $(2,2)$, *then find the shortest path for node* $(7,3)$ *to the existing route, which is* $(2,2) \Rightarrow (7,2) \Rightarrow (7,3)$. *Since the left nodes share the same distance, choose node* $(6,7)$ *next, whose path is* $(6,2) \Rightarrow (6,7)$. *Finally apply the nodes* $(9,4)$ *and* $(7,6)$, *whose paths are* $(6,4) \Rightarrow (9,4)$ *and* $(6,6) \Rightarrow (7,6)$. *The resulting routing is shown in Figure 2.9.*

The multicasting time $TIME_{DIST_{2DM}} = 14 > D_{max} + 1$,

The number of links $LINK_{DIST_{2DM}} = 19 < x_i + \sum_{j=1}^{i} y_j$.

Compare the final route with that from example 2.3:

$TIME_{VH_{2DM}} = TIME_{DIST_{2DM}}$,

$LINK_{VH_{2DM}} > LINK_{DIST_{2DM}}$.

The more general relationships between these algorithms will be discussed in the next section.

2.2.3 Comparisons of the VH_{2DM} and $DIST_{2DM}$ Algorithms

The examples above show some ideas about the relationship of the multicasting time and the total number of links. The two propositions below establish the relationships between the VH_{2DM} Algorithm and the $DIST_{2DM}$ Algorithm.

Proposition 2.3. *In the 2-dimensional mesh network, the multicasting time in the* VH_{2DM} *algorithm is equal to or less than the multicasting time in the* $DIST_{2DM}$ *algorithm.*

$$TIME_{VH_{2DM}} \leq TIME_{DIST_{2DM}}$$

Proof. Proposition 2.2 shows that the multicasting time for the VH_{2DM} algorithm is always the minimum possible value. Then $TIME_{VH_{2DM}} \leq TIME_A$ for any algorithm A. Hence $TIME_{VH_{2DM}} \leq TIME_{DIST_{2DM}}$. $\square$

The relationship for the number of links is displayed in the following:

Proposition 2.4. *In the 2-dimensional mesh network, the number of links in the VH_{2DM} algorithm is equal to or greater than the number of links in the $DIST_{2DM}$ algorithm.*

$$LINK_{VH_{2DM}} \geq LINK_{DIST_{2DM}}$$

Proof. Assume the VH multicasting tree is constructed. Take node (x_1, y_1) for which distance is at a minimum. In the $DIST$ algorithm, node (x_1, y_1) will be constructed in the same way as in the VH algorithm. Take the next closest node (x_2, y_2). In the VH algorithm, only add y_2 edges to connect (x_2, y_2) to the multicasting tree. On the other hand, the $DIST$ algorithm adds the minimum number of edges necessary from (x_2, y_2) to the same tree, which is either equal to or smaller than the value y_2. Then add node (x_3, y_3) and continue in this way until every destination is in the multicasting tree. Thus, $LINK_{VH_{2DM}} \geq LINK_{DIST_{2DM}}$. $\square$

In the 2-dimensional mesh network, the VH_{2DM} algorithm guarantees that the message will be delivered in the minimum amount of time, but will generate more traffic. The $DIST_{2DM}$ algorithm can reduce the total amount of traffic at the price of spending more multicasting time. In a real time network, in order to deliver the message in the least amount of time, the VH_{2DM} applies. If the traffic is considered more important than the multicasting time, it is better to use the $DIST_{2DM}$ algorithm.

2.3 2-Dimensional Torus Network

A 2-dimensional torus is a 2-dimensional mesh with wraparound links and each node has exactly 4 neighbors. The symmetry of the torus network leads to a more balanced utilization of communication links than that of the mesh topology

In an $m \times n$ 2D torus network ($2DT$), we assume multicasting requires sending a message from the source $s(0, 0)$ to a set of destinations $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$, where $0 \leq x_i \leq m-1$ and $0 \leq y_i \leq n-1$. The destinations in 2DM always follows the positive direction going back to the source, while the destinations in 2DT may follow in either positive or negative direction, depending on the position of these nodes. The 2DT graph can be divided into four sections, and each section forms a mesh graph. Each node (x_j, y_j) belongs to one of the four sections below.

- section 1: $x_j \leq \lfloor \frac{m}{2} \rfloor$ and $y_j \leq \lfloor \frac{n}{2} \rfloor$,

- section 2: $x_j > \lfloor \frac{m}{2} \rfloor$ and $y_j \leq \lfloor \frac{n}{2} \rfloor$,

- section 3: $x_j \leq \lfloor \frac{m}{2} \rfloor$ and $y_j > \lfloor \frac{n}{2} \rfloor$,

- section 4: $x_j > \lfloor \frac{m}{2} \rfloor$ and $y_j > \lfloor \frac{n}{2} \rfloor$.

After dividing the 2DT into four sections, the 2DT becomes four sub-2DM problems, the routing paths follow in a different direction, based on which section the destinations are in. In section 1, all the destinations follow in a positive direction for both vertical and horizontal dimensions, and go back to source $s(0, 0)$. In section 2, all the destinations go toward node $(m-1, 0)$, then go back to source $s(0, 0)$ following the positive direction in the vertical dimension and negative direction in the horizontal dimension. Similarly, in section 3, all of the destinations go to node $(0, n-1)$ first, then go back to source $(0, 0)$ following the negative direction in the vertical dimension and positive direction in the horizontal

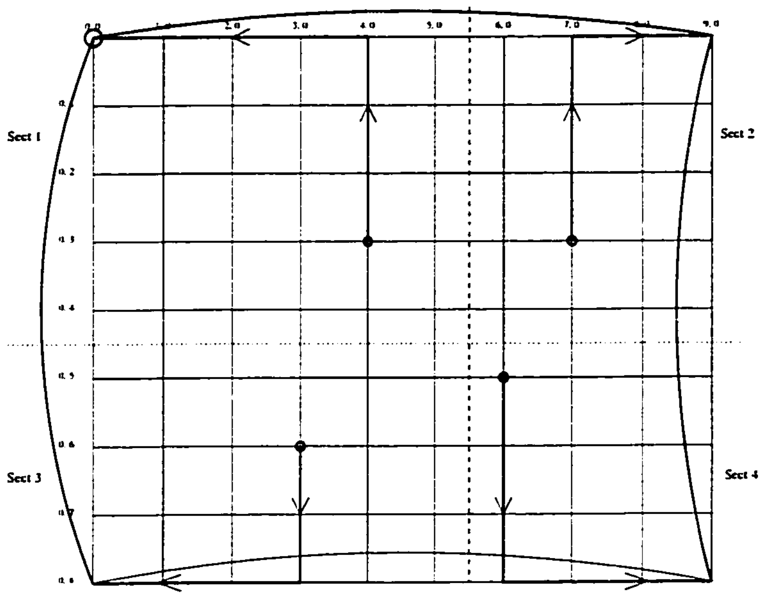


Figure 2.10: 2-Dimensional torus network

dimension. All the destinations in section 4 reach the node $(m-1, n-1)$ first , then go back to source $(0, 0)$ using the negative direction for both vertical and horizontal dimensions. See illustration in Figure 2.10.

Definition 2.4. *In a 2DT* $(m \times n)$*, a set of destinations are* $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$*, where* $0 \leq x_i \leq m-1$ *and* $0 \leq y_i \leq n-1$*. By definition,* D_{max} *is the maximum value distance for each destination node. The* x_{max} *and* y_{max} *are defined as,*

$x_{max} = max\{x_1, \ldots, x_i\}$

$y_{max} = max\{y_1, \ldots, y_i\}$

2.3.1 VH Algorithm in 2-Dimensional Torus Network

2DT consists of four 2DM problems. Apply the VH_{2DM} to each section.

Algorithm 2.3 (The VH_{2DT} Algorithm). *In 2D torus network* $(m \times n)$*, given a source s(0, 0) and a set of destinations* $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$*, where* $0 \leq x_i \leq m-1$ *and*

$0 \leq y_i \leq n-1$. *Multicasting requires sending a message from s to all* (x_i, y_i)*s.*

1. *Assign distances* (D_j) *for all destinations.*

   - *If the node is in section 1,* $D_j = x_j + y_j$, *for* $1 \leq j \leq i$,
   - *If the node is in section 2,* $D_j = (m - x_j) + y_j$, *for* $1 \leq j \leq i$,
   - *If the node is in section 3,* $D_j = x_j + (n - y_j)$, *for* $1 \leq j \leq i$,
   - *If the node is in section 4,* $D_j = (m - x_j) + (n - y_j)$, *for* $1 \leq j \leq i$.

2. *Construct the multicasting tree.*

   - *Start from the destination with the maximum x value.*
   - *If the node is in section 1,*
     - *Follow the positive direction in the vertical dimension,*
     - *Follow the positive direction in the horizontal dimension.*
   - *If the node is in section 2,*
     - *Follow the positive direction in the vertical dimension,*
     - *Follow the negative direction in the horizontal dimension.*
   - *If the node is in section 3,*
     - *Follow the negative direction in the vertical dimension,*
     - *Follow the positive direction in the horizontal dimension.*
   - *If the node is in section 4,*
     - *Follow the negative direction in the vertical dimension,*
     - *Follow the negative direction in the horizontal dimension.*

3. *All the nodes in section 2, 3, and 4 will eventually go back to the source via the wraparound links.*

4. Repeat the process for all destination nodes.

5. If more than one node has the x value, select one at random.

In the VH_{2DM} Algorithm, the maximum multicasting time is D_{max}+1 when more than one node has the distance value of D_{max}, while in VH_{2DT}, if these nodes are in section 2 and 3, they can be reached via a wraparound link; if these nodes are in section 4, they can be reached via two wraparound links from the source node, which needs two more time units to transmit the message. The multicasting time now satisfied,

$$D_{max} \leq TIME_{VH_{2DT}} \leq D_{max} + 3 \tag{2.5}$$

The total number of links also depends on the pattern of destinations. As in the VH_{2DM} algorithm, if the x value of each node is unique, the number of links will count every y value. If some nodes have the same x value, then only the maximum y value among these nodes is counted. The following procedure can be used to calculate the sum of total links in the VH_{2DT} algorithm.

Procedure 2.2. *The sum of total links.*

```
$\sum_{2DT} = 0;$
$X_{max1} = X_{max3} = 0;$
$X_{max2} = X_{max4} = m - 1;$
$Y_1 = Y_2 = Y_3 = Y_4 = 0;$
do
  take node $(x_j, y_j)$
  if node $(x_j, y_j)$ is in section 1,
      if $x_j > X_{max1}$
```

```
        $X_{max1} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ in the sum, $Y_1 = Y_1 + y_j$.
    else if $x_j = x_{j+1} = \cdots = x_p$
        then include only $y_p$ in the sum, $Y_1 = Y_1 + y_p$.
else if node $(x_j, y_j)$ is in section 2,
    if $x_j < X_{max2}$
        $X_{max2} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ in the sum, $Y_2 = Y_2 + y_j$.
    else if $x_j = x_{j+1} = \cdots = x_p$
        then include only $y_p$ in the sum, $Y_2 = Y_2 + y_p$.
else if node $(x_j, y_j)$ is in section 3,
    if $x_j > X_{max3}$
        $X_{max3} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ in the sum, $Y_3 = Y_3 + (n - y_j)$.
    else if $x_j = x_{j+1} = \cdots = x_p$
        then include only $y_p$ in the sum, $Y_3 = Y_3 + (n - y_p)$.
else if node $(x_j, y_j)$ is in section 4,
    if $x_j < X_{max4}$
        $X_{max4} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ in the sum, $Y_4 = Y_4 + (n - y_j)$.
        else if $x_j = x_{j+1} = \cdots = x_p$
            then include only $y_p$ in the sum, $Y_4 = Y_4 + (n - y_p)$.
    enddo
```

$$\textstyle\sum_{2DT} = \sum \begin{cases} X_{max1} + Y_1, & \textit{if any node } (x_j, y_j) \textit{ is in section 1;} \\ m - X_{max2} + Y_2, & \textit{if any node } (x_j, y_j) \textit{ is in section 2;} \\ X_{max3} + 1 + Y_3, & \textit{if any node } (x_j, y_j) \textit{ is in section 3;} \\ m - X_{max4} + 1 + Y_4, & \textit{if any node } (x_j, y_j) \textit{ is in section 4.} \end{cases}$$

output $\sum_{2DT}$.

The total number of links can be described as

$$LINK_{VH_{2DT}} = \sum. \tag{2.6}$$

where $\sum$ *represents the output of Procedure 2.2.*

The minimum value of $LINK_{VH_{2DT}}$ will be obtained in the case where all nodes have the same x value, and they are in one section, then $LINK_{VH_{2DT}} = x_i + y_i$ in the case where all nodes are in section one. The maximum value of $LINK_{VH_{2DT}}$ will be obtained in the case where all x_i's are mutually different, then the value $LINK_{VH_{2DM}}$ will count every link.

The multicasting time in 2DM and 2DT have a similar relationship, except the upper bound of 2DT is $D_{max} + 3$ instead of $D_{max} + 1$, since the torus needs two more time units to reach the nodes in section 4. Note that the D_{max} in 2DT is smaller than the D_{max} value in 2DM for the same set of destination nodes.

To illustrate the VH_{2DT} algorithm, consider the same data as in the 2DM network.

Example 2.6. *In a 2DT network* (10×9), *a message is sent from node* $S(0,0)$ *to a set of nodes* $\{(2,2), (4,3), (5,4), (6,6), (7,6)\}$. *By using the* VH_{2DT} *Algorithm, node* $(5,4)$ *has the* *maximum distance value* $D_{max} = 5 + 4 = 9$, *which is smaller that the* D_{max} *value for 2DM in example 2.2. Nodes* $(2,2), (4,3), (5,4)$ *are in section 1, and their multicasting paths follow in a positive direction for both vertical and horizontal dimensions. Nodes* $(6,6)$ *and* $(7,6)$ *are in section 4, and their paths follow in a negative direction for both vertical and horizontal dimensions toward node* $(9,8)$, *followed by node* $(9,0)$ *and* $(0,0)$ *via wraparound links. The multicasting time* $TIME_{VH_{2DT}} = D_{max} = 9$, *and the number of links* $LINK_{VH_{2DT}} = 23$. *The resulting route is shown in Figure 2.11.*

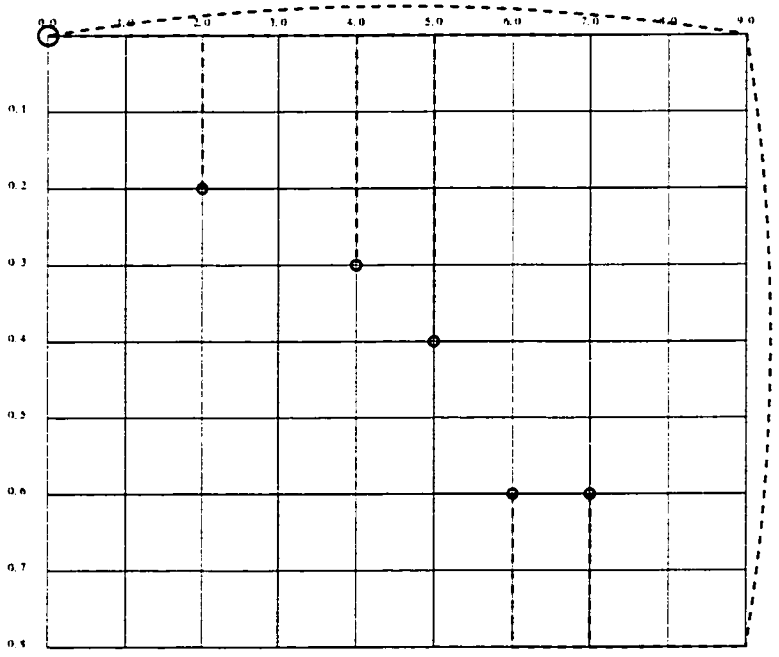


Figure 2.11: Multicasting route for the VH_{2DT} algorithm

Example 2.7. *In a 2DT network* (10×9), *a message is sent from node* $S(0,0)$ *to a set of nodes* $\{(2,2),(6,7),(7,3),(7,6),(9,4)\}$. *When applying the* VH_{2DT} *Algorithm, the maximum distance is* $D_{max} = 6$, *and three nodes,* $(6,7)$, $(7,3)$, *and* $(7,6)$, *have the distance value* D_{max}. *Every destination follows in the vertical dimension first, then turns into the horizontal dimension. Node* $(2,2)$ *is in section 1, and its path follows in the positive direction*

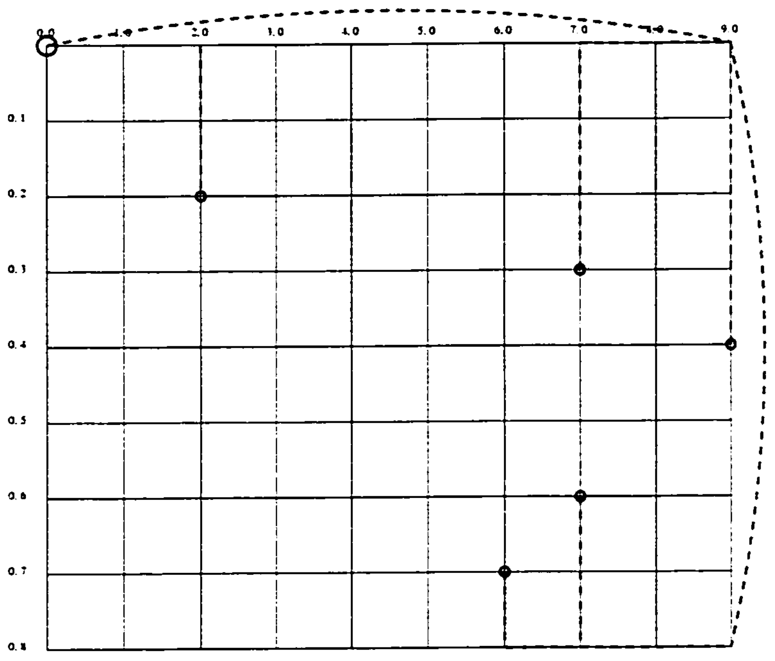


Figure 2.12: Routing scheme for the VH_{2DT} algorithm

for both vertical and horizontal dimensions. Nodes (7, 3) *and* (9, 4) *are in section 2, and their paths follow in the positive direction for the vertical dimension, then in the negative direction for the horizontal dimension. Nodes* (6, 7) *and* (7, 6) *are in section 4, their paths follow in the negative direction for both vertical and horizontal dimensions. The multicasting time* $TIME_{VH} = D_{max} + 1 = 7$*, and the number of links* $LINK_{VH} = 21$*. The resulting routing is shown in Figure 2.12.*

2.3.2 *DIST* Algorithm in the 2-Dimensional Torus Network

The $DIST$ algorithm can also be applied to the 2-dimensional torus network since it contains four 2DM subgraphs.

Algorithm 2.4 (The $DIST_{2DT}$ Algorithm). *In a 2DT network* $(m \times n)$*, given a source s(0, 0) and a set of destinations* $\{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$*, where* $0 \leq x_i \leq m - 1$ *and* $0 \leq y_i \leq n - 1$*. Multicasting requires sending a message from s to all* (x_j, y_j)*s.*

- *Assign distances (D_j) for all destination nodes.*
  - *If the node is in section 1, $D_j = x_j + y_j$, for $1 \le j \le i$,*
  - *If the node is in section 2, $D_j = (m - x_j) + y_j$, for $1 \le j \le i$,*
  - *If the node is in section 3, $D_j = x_j + (n - y_j)$, for $1 \le j \le i$,*
  - *If the node is in section 4, $D_j = (m - x_j) + (n - y_j)$, for $1 \le j \le i$.*
- *Build the routes to connect all the destinations to the source: start from the destination with the minimal distance, and find the shortest path to the existing multicasting tree in its own section.*
  - *Start from the destination with the minimal distance.*
  - *Find the shortest path to the existing multicasting tree in its own section.*
  - *Repeat the procedure until every destination is included in the multicasting tree.*
- *If more than one node has the same distance in a section, alternatively take (x_j, y_j) for which x_j is the minimum possible value and take (x_k, y_k) for which x_k is the maximum possible value in its section.*

The multicasting time in $DIST_{2DT}$ has the same relationship as in $DIST_{2DM}$, however the value of D_{max} is much smaller than the D_{max} of the 2DM network.

$$D_{max} \le TIME_{DIST_{2DT}} < min\{m + \frac{n}{2}, n + \frac{m}{2}, 3D_{max}\} \tag{2.7}$$

The total number of links depends on the pattern of destinations. In addition, when counting the number of links, they are counted within each section, and finally added all together. The path in $DIST_{2DT}$ can not cross different sections.

$$LINK_{DIST_{2DT}} \leq \sum \begin{cases} x_{max1} + \sum y_{j1}, & \text{for } (x_j, y_j) \text{ in section 1;} \\ m - x_{max2} + \sum y_{j2}, & \text{for } (x_j, y_j) \text{ in section 2;} \\ x_{max3} + 1 + \sum(n - y_{j3} - 1), & \text{for } (x_j, y_j) \text{ in section 3;} \\ m - x_{max4} + 1 + \sum(n - y_{j4} - 1), & \text{for } (x_j, y_j) \text{ in section 4;} \end{cases} \tag{2.8}$$

To illustrate how the $DIST_{2DT}$ algorithm works, let us consider the examples used in the 2DM network.

Example 2.8. *In 2DT network* (10×9), *a message is sent from node* $s(0,0)$ *to a set of nodes* $\{(2,2), (4,3), (5,4), (6,6), (7,6)\}$. *By using the* $DIST_{2DT}$ *Algorithm, node* $(5,4)$ *has the maximum distance value* $D_{max} = 5 + 4 = 9$. *Nodes* $(2,2)$ $(4,3)$ *and* $(5,4)$ *are in section 1. First find the shortest path for node* $(2,2)$, *then find the path for node* $(4,3)$, *followed by node* $(5,4)$. *The shortest path is*

$(0,0) \Rightarrow (2,0) \Rightarrow (2,2) \Rightarrow (4,2) \Rightarrow (4,3) \Rightarrow (5,3) \Rightarrow (5,4)$;

Nodes $(6,6)$ *and* $(7,6)$ *are in section 4, so apply to node* $(7,6)$ *first, then consider node* $(6,6)$. *The path is*

$(0,0) \Rightarrow (9,0) \Rightarrow (9,8) \Rightarrow (7,8) \Rightarrow (7,6) \Rightarrow (6,6)$.

The resulting route is shown in Figure 2.13.

The multicasting time is $TIME_{VH_{2DT}} = D_{max} = 9$,

The number of links is $LINK_{VH_{2DT}} = 16$.

Compare the results with the ones obtained from example 2.6:

$TIME_{VH} = TIME_{Distance}$

$LINK_{VH} > LINK_{Distance}$

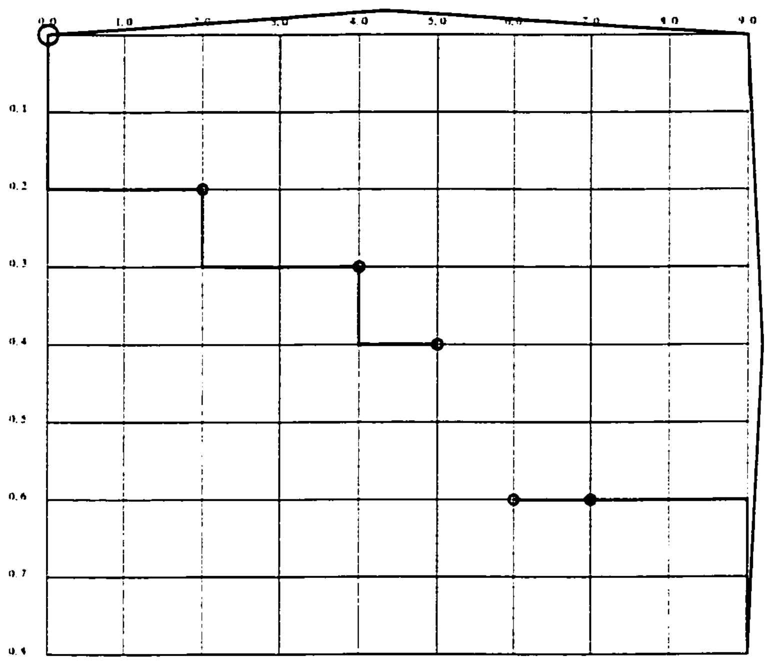


Figure 2.13: Multicast routing for $DIST_{2DT}$ algorithm

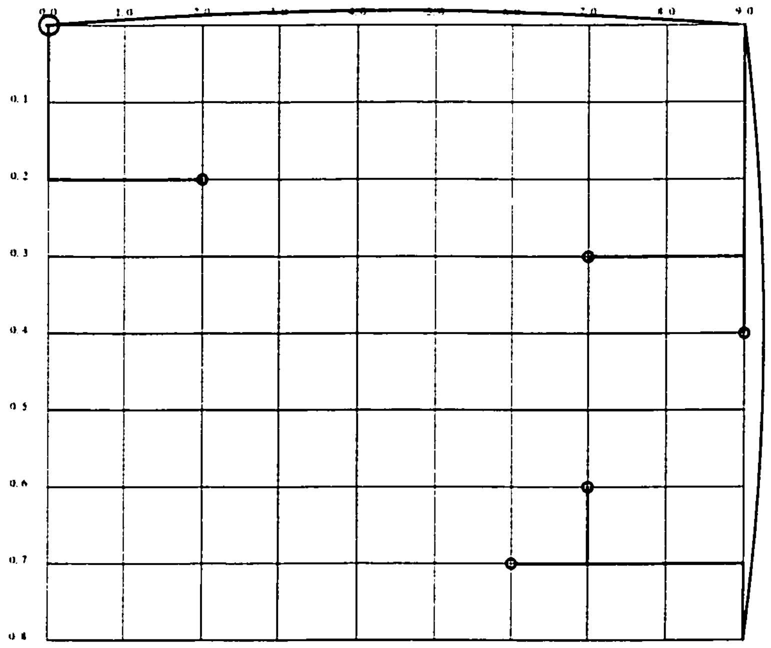


Figure 2.14: Routing scheme for $DIST_{2DT}$ algorithm

Example 2.9. *In the 2DT network* (10×9), *a message is sent from node* $s(0,0)$ *to a set of nodes* $\{(2,2),(6,7),(7,3),(7,6),(9,4)\}$. *When applying the* $DIST_{2DT}$ *Algorithm, the maximum distance value* $D_{max} = 6$, *and three nodes,* $(6,7)$, $(7,3)$, *and* $(7,6)$, *have the value* D_{max}. *Node* $(2,2)$ *is in section 1: find the shortest path back to the source node* $s(0,0)$. *Nodes* $(7,3)$ *and* $(9,4)$ *are in section 2: find the shortest path for node* $(9,4)$ *first, then apply to node* $(7,3)$ *within section 2. Nodes* $(6,7)$ *and* $(7,6)$ *are in section 4, since they have the same distance value: choose node* $(6,7)$ *first, and then find the shortest path for node* $(7,6)$ *in section 4. The resulting route is shown in Figure 2.14.*

The multicasting time $TIME_{VH} = D_{max} + 1 = 7$,

The number of links $LINK_{VH} = 17$.

Compare the results with those found in example 2.7:

$TIME_{VH} < TIME_{Distance}$

$LINK_{VH} > LINK_{Distance}$.

2.3.3 Comparisons of the VH_{2DT} and $DIST_{2DT}$ Algorithms

In the 2DT network, the relationship between VH_{2DT} and $DIST_{2DT}$ is the same as in 2DM, as previously discussed that the 2-dimensional torus consists of four 2DM subgraphs.

Proposition 2.5. *In the 2-dimensional torus network, the multicasting time of the* VH_{2DT} *algorithm is equal to or less than the multicasting time of the* $DIST_{2DT}$ *algorithm.*

$$TIME_{VH_{2DT}} \leq TIME_{DIST_{2DT}}$$

Proposition 2.6. *In the 2-dimensional torus network, the number of links in the* VH_{2DT} *algorithm is equal to or greater than the number of links in the* $DIST_{2DT}$ *algorithm.*

$$LINK_{VH_{2DT}} \geq LINK_{DIST_{2DT}}$$

2.4 3-Dimensional Mesh Network

2.4.1 VH Algorithm in 3-Dimensional Mesh Network

Now we extend the algorithms to the 3-dimensional mesh network ($3DM$). In the 2DM, the route follows in the vertical dimension first, then turns into the horizontal dimension. We apply the similar dimension-ordered strategy to the $3DM$. Routing proceeds on the depth dimension (say z dimension) first, then goes to the horizontal dimension (say y dimension), and finally follows the vertical dimension (say x dimension) back to the source $s(0,0)$.

Algorithm 2.5 (The VH_{3DM} Algorithm). *In a 3DM network* $(m \times n \times p)$, *given a source s(0, 0, 0) and a set of destinations* $\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_i, y_i, z_i)\}$, *where* $0 \leq x_i \leq m-1$, $0 \leq y_i \leq n-1$, *and* $0 \leq z_i \leq p-1$. *Multicasting requires a message to be sent from s to all* (x_i, y_i, z_i)*s.*

1. *Assign the distances* (D_j) *for all destinations* $(D_j = x_j + y_j + z_j)$, *where* $1 \leq j \leq i$.
2. *Construct the multicasting tree.*
   - *Start from the destination with value* x_{max} *and* y_{max}, *expand the path backward.*
   - *Follow the z dimension until the node* $(x_j, y_j, 0)$.
   - *Follow the y dimension until the node* $(x_j, 0, 0)$.
   - *Along the x dimension until returning to the source* $s(0,0,0)$.
   - *Repeat the process for all destination nodes.*
3. *If more than one node has the same x value, first choose the node that has the greatest y value.*
4. *If more than one node has the same x and y value, choose the one that has the greatest z value.*

In the VH_{3DM} Algorithm, because each node follows the shortest path, the total multicasting time is at a minimum. Assume the set of destinations is $\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_i, y_i, z_i)\}$, where $x_1 \leq x_2 \leq \cdots \leq x_i$, and if nodes $\{(x_j, y_j, z_j), \ldots, (x_k, y_k, z_k)\}$ have the same x value, say $x_j = x_{j+1} = \cdots = x_k$, and assume $y_j \leq y_{j+1} \leq \cdots \leq y_k$. Furthermore, if these nodes have the same x and y value, say $x_j = x_{j+1} = \cdots = x_k$ and $y_j = y_{j+1} = \cdots = y_k$, then $z_j \leq z_{j+1} \leq \cdots \leq z_k$. If only one destination has the maximum distance value, the multicasting time is equal to the value of maximum distance D_{max}. If more than one destination has the maximum distance value, the algorithm may need two more time units in order to finish multicasting, since any node has three directions (x, y, or z dimension) to start with, it will possibly cause two time units delay. However, the multicasting time in the VH_{3DM} algorithm is at a minimum.

Proposition 2.7. *The VH_{3DM} Algorithm always generates the minimal multicasting time.*

$$D_{max} \leq TIME_{VH_{3DM}} \leq D_{max} + 2, \tag{2.9}$$

where $D_{max} = max\{x_1 + y_1 + z_1, \ldots, x_i + y_i + z_i\}$.

The total number of links also depends on the pattern of destinations. If the x and y value of each node is unique, the number of links will count every z value. If some nodes have the same x and y value, then only the maximum z value among these nodes is counted. Consequently, if the x value of each node is unique, the number of links will count every y value. If some nodes have the same x value, then only the maximum y value among these nodes is counted. The total number of links satisfies $LINK_{VH_{3DM}} \leq x_i + \sum_{j=1}^{i} y_j + \sum_{k=1}^{i} z_k$. To calculate the exact sum of total y value, the following procedure can be used.

Procedure 2.3. *The sum of total links in the $3DM$ network.*

$\sum_{3DM} = 0;$

```
do
    take node $(x_j, y_j, z_j)$
    if $x_j$ is unique,
        then include $y_j$ and $z_J$ in the sum, $\sum_{3DM} = \sum_{3DM} + y_j + z_j$.
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $\sum_{3DM} = \sum_{3DM} + y_p + z_p$.
        otherwise
            include every $y_j$ and $z_j$ in the sum, $\sum_{3DM} = \sum_{3DM} + y_j + z_j$.
enddo
output $\sum_{3DM}$.
```

The total number of links can be described as

$$LINK_{VH_{3DM}} = x_i + \sum . \tag{2.10}$$

where x_i is the maximum x value and $\sum$ is the output of Procedure 2.3.

The maximum value of $LINK_{VH_{3DM}}$ will be obtained in the case where all x_i's are mutually different, then the value $LINK_{VH_{3DM}} = x_i + \sum_{j=1}^{i} y_j + \sum_{k=1}^{i} z_k$. The minimum value of $LINK_{VH_{3DM}}$ will be obtained in the case where all nodes have the same x and y value, say $x_1 = x_2 = \cdots = x_i$ and $y_1 = y_2 = \cdots = y_i$, then $LINK_{VH_{3DM}} = x_i + y_i + z_i$. Thus, the $LINK_{VH_{3DM}}$ satisfies: $x_i + y_i + z_i \leq LINK_{VH_{3DM}} \leq x_i + \sum_{j=1}^{i} y_j + \sum_{k=1}^{i} z_k$.

Here is an illustration of how the VH Algorithm in the 3DM works.

Example 2.10. *In the 3D torus network* $(10 \times 9 \times 4)$, *the message is sent from node* $s(0,0,0)$ *to a set of nodes* $\{(2,1,0), (4,3,0), (3,4,0), (8,3,0), (8,7,0), (9,0,3), (9,7,1), (9,7,2)\}$. *By using the VH_{3DM} Algorithm, node* $(5,4,1)$ *has the maximum distance value* $D_{max} = 9 +$

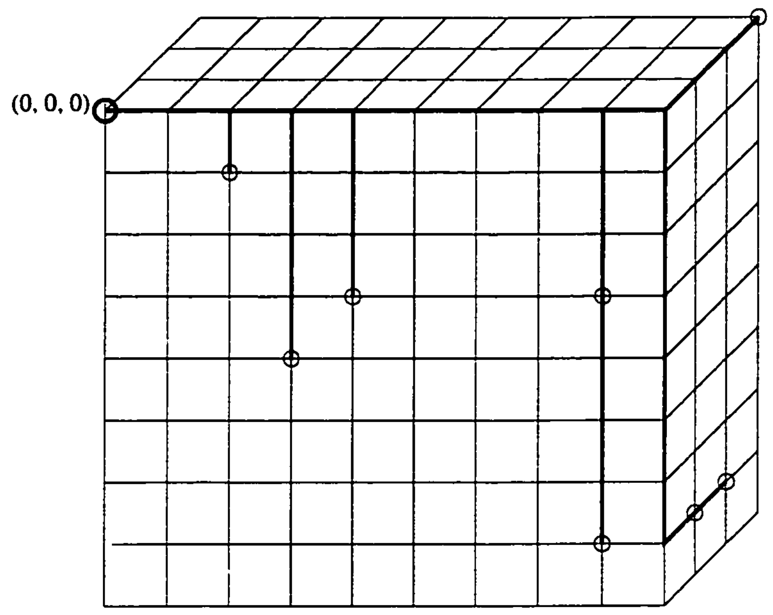


Figure 2.15: Multicast routing in the VH_{3DM} algorithm

$7+2=18$. *Start with node* $(9,7,2)$ *to find the shortest path that follows the z dimension first, then along the y dimension and x dimension until reaching the source* $s(0,0,0,)$. *The possible path is:* $(9,7,2) \Rightarrow (9,7,0) \Rightarrow (9,0,0) \Rightarrow (0,0,0)$. *Next, find the shortest path for node* $(8,7,0)$, *which is:* $(8,7,0) \Rightarrow (8,0,0) \Rightarrow (0,0,0)$. *Then, find the shortest path for node* $(9,0,3)$, $(4,3,0)$, $(3,4,0)$, *and* $(2,1,0)$. *The resulting routing is shown in Figure 2.15.*

The multicasting time $TIME_{VH_{3DM}} = D_{max} = 18$,

The output of procedure 2.3 is $\sum = 27$ *and the maximum x value is* $x_i = 9$. *So that the number of links* $LINK_{VH_{3DM}} = x_i + \sum = 36$.

2.4.2 *DIST* Algorithm in the 3-Dimensional Mesh Network

The *DIST* algorithm always finds the shortest path connected to the existing multicasting tree for each node. Hence, it can reduce the total traffic. However, the multicasting time will not be minimal because the shortest path to the existing tree may not be the shortest path between the source and the destination nodes. Therefore, more transmission time may be needed when using the shortest distance approach.

Algorithm 2.6 (The $DIST_{3DM}$ Algorithm). *In a 3DM network ($m \times n \times p$), given a source s(0, 0, 0) and a set of destinations $\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_i, y_i, z_i)\}$, where $0 \leq x_i \leq m-1$, $0 \leq y_i \leq n-1$, and $0 \leq z_i \leq p-1$. Multicasting requires a message to be sent from s to all (x_i, y_i, z_i)s.*

1. *Assign distances (D_j) for all destinations ($D_j = x_j + y_j + z_j$, where $1 \leq j \leq i$).*

2. *Build the routes to connect all destinations to the source:*

   - *Start from the destination with the shortest distance.*
   - *Find the shortest path to the existing multicasting tree.*
   - *Repeat the process in distance ascending order until every destination node is included in the multicasting tree.*

3. *If more than one node has the same distance, alternatively take (x_j, y_j, z_j) for which x_j is the minimum possible value and take (x_k, y_k, z_k) for which x_k is the maximum possible value.*

   - *If these nodes have the same x value, compare the y values and follow the same order.*

The multicasting time depends on the location of destination nodes. When several nodes have the maximum distance value D_{max}, the multicasting time may be close to the value of D_{max}, or also close to the value of $5D_{max}$, if the nodes with the value of D_{max} are chosen randomly.

$$D_{max} \leq TIME_{DIST_{3DM}} < min\{2m + 2n + p - 5, 5D_{max}\}, \qquad (2.11)$$

where $D_{max} = max\{x_1 + y_1 + z_1, \ldots, x_i + y_i + z_i\}$.

The total number of links various and depends on the pattern of destinations, but the upper bound for the total is given by the expression below. This bound is achieved when the shortest path from the current node (x_j, y_j, z_j) to the existing tree is the shortest path from node (x_j, y_j, z_j) to node $(x_{j-1}, y_{j-1}, z_{j-1})$.

Thus, $LINK_{DIST_{3DM}} \leq$

$(x_1+y_1+z_1)+(|x_2-x_1|+|y_2-y_1|+|z_2-z_1|)+\cdots+(|x_j-x_{j-1}|+|y_j-y_{j-1}|+|z_j-z_{j-1}|)$.

This means that

$$LINK_{DIST_{3DM}} \leq (x_1 + y_1 + z_1) + \sum_{j=2}^{i}(|x_j - x_{j-1}| + |y_j - y_{j-1}| + |z_j - z_{j-1}|) \qquad (2.12)$$

The minimum value of $LINK_{DIST_{3DM}}$ will be obtained in the case where all nodes have the same x and y value, say $x_1 = \cdots = x_i$ and $y_1 = \cdots = y_i$, then $LINK_{DIST_{3DM}} = x_1 + y_1 + \sum_{j=2}^{i}(z_j - z_{j-1}) = x_i + y_i + z_i$. The maximum value of $LINK_{DIST_{3DM}}$ can be reached when all x_is are mutually different, then $LINK_{DIST_{3DM}} = x_i + \sum_{j=1}^{i} y_j + \sum_{j=1}^{i} z_j$.

To illustrate how the algorithm works, let us consider an example below.

Example 2.11. *Consider a 3DT network* $(10 \times 9 \times 4)$, *the message is sent from node* $s(0,0,0)$ *to a set of nodes* $\{(2,1,0), (4,3,0), (3,4,0), (8,7,0), (9,0,3), (9,7,2)\}$. *By using the* $DIST_{3DM}$ *Algorithm, node* $(5,4,1)$ *has the maximum distance value* $D_{max} = 9 + 7 + 2 = 18$, *and node* $(2,1,0)$ *has the minimum distance vakue. First, find the shortest path for node* $(2,1,0))$, *which is* $(0,0,0) \Rightarrow (2,0,0) \Rightarrow (2,1,0)$. *Nodes* $(4,3,0)$ *and* $(3,4,0)$ *have the same distance value. Select nodes in the order of* $(3,4,0)$ *and* $(4,3,0)$, *the paths are* $(2,1,0) \Rightarrow (3,1,0) \Rightarrow (3,4,0)$ *and* $(3,3,0) \Rightarrow (4,3,0)$. *Afterward find the shortest path for nodes* $(9,0,3)$, $(8,7,0)$, *and* $(9,7,2)$. *The final routing scheme is shown in Figure 2.16(1). The multicasting time* $TIME_{DIST_{3DM}} = 18$, *The number of links* $LINK_{DIST_{3DM}} = 28$.

When finding paths for nodes $(4,3,0)$ *and* $(3,4,0)$, *if nodes in the order of* $(4,3,0)$ *and* $(3,4,0)$ *are selected, then the final path becomes Figure 2.16(2).*

The multicasting time $TIME_{DIST_{3DM}} = 20$,

The number of links $LINK_{DIST_{3DM}} = 28$.

When several nodes have the same distance value whenever the nodes have the maximum D_{max} *or not, the resulting multicasting times and total traffic may vary.*

Comparing the parameters, time and traffic, with those from the example 2.10:

$TIME_{VH_{3DM}} \leq TIME_{DIST_{3DM}}$

$LINK_{VH_{3DM}} > LINK_{DIST_{3DM}}$

2.4.3 Comparisons of the VH_{3DM} and the $DIST_{3DM}$ Algorithms

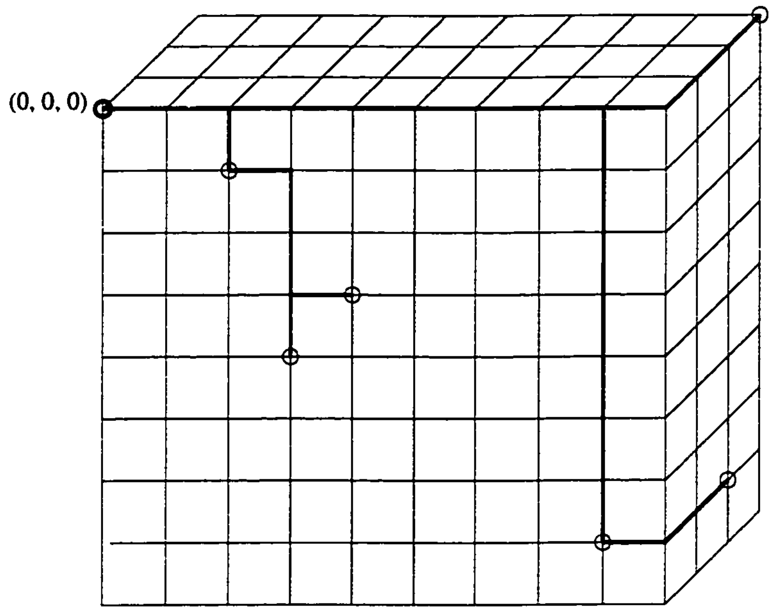
As in the 2DM network, all nodes in the VH_{3DM} algorithm always follow the shortest path independently, and this guarantees that the message will be delivered in the minimum multicasting time. However, the VH_{3DM} algoritm generates more total traffic than $DIST_{3DM}$ algorithm. On the other hand, the $DIST_{3DM}$ algorithm follows the shortest path connected to the existing tree, which can reduce the total traffic, for a price of extra multicasting time.

Proposition 2.8. *In the 3-dimensional mesh network, the multicasting time in the* VH_{3DM} *algorithm is equal to or less than the multicasting time in the* $DIST_{3DM}$ *algorithm.*

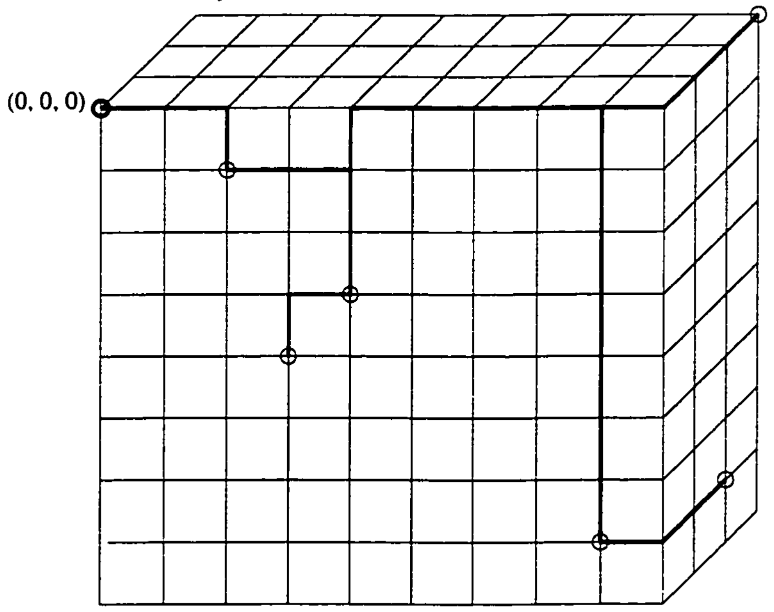
$$TIME_{VH_{3DM}} \leq TIME_{DIST_{3DM}}$$

Proposition 2.9. *In the 3-dimensional mesh network, the number of links in the* VH_{3DM} *algorithm is equal to or greater than the number of links in the* $DIST_{3DM}$ *algorithm.*

$$LINK_{VH_{3DM}} \geq LINK_{DIST_{3DM}}$$



(1) find path in the order of (3, 4, 0) and (4, 3, 0)



(2) find path in the order of (4, 3, 0) and (3, 4, 0)

Figure 2.16: Multicast routing using the $DIST_{3DM}$ algorithm

Because the proofs of these propositions are similar to the proofs of proposition 2.3 and 2.4, they are omitted.

2.5 3-Dimensional Torus Network

The 3-dimensional torus (3DT) has the same properties as in the 2DT. The 3DT can be seen as a 3DM with wraparound links in every dimension. In general, assuming a $3DT$ network $(m \times n \times p)$, the multicasting requires a message to be sent from the source $s(0,0,0)$ to a set of destinations $\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_i, y_i, z_i)\}$, where $0 \leq x_i \leq m-1$, $0 \leq y_i \leq n-1$, and $0 \leq z_i \leq p-1$. Unlike the 2DT network, which consists of four 2DM subgraphs, the 3DT network can be divided into eight 3DM subgraphs, and it becomes a 3-dimensional mesh problem. Each node (x_j, y_j, z_j) can be distributed into one of these eight sections.

- section 1: $x_j \leq \lfloor \frac{m}{2} \rfloor$, $y_j \leq \lfloor \frac{n}{2} \rfloor$, and $z_j \leq \lfloor \frac{p}{2} \rfloor$,

- section 2: $x_j > \lfloor \frac{m}{2} \rfloor$, $y_j \leq \lfloor \frac{n}{2} \rfloor$, and $z_j \leq \lfloor \frac{p}{2} \rfloor$,

- section 3: $x_j \leq \lfloor \frac{m}{2} \rfloor$, $y_j > \lfloor \frac{n}{2} \rfloor$, and $z_j \leq \lfloor \frac{p}{2} \rfloor$,

- section 4: $x_j > \lfloor \frac{m}{2} \rfloor$, $y_j > \lfloor \frac{n}{2} \rfloor$, and $z_j \leq \lfloor \frac{p}{2} \rfloor$,

- section 5: $x_j \leq \lfloor \frac{m}{2} \rfloor$, $y_j \leq \lfloor \frac{n}{2} \rfloor$, and $z_j > \lfloor \frac{p}{2} \rfloor$,

- section 6: $x_j > \lfloor \frac{m}{2} \rfloor$, $y_j \leq \lfloor \frac{n}{2} \rfloor$, and $z_j > \lfloor \frac{p}{2} \rfloor$,

- section 7: $x_j \leq \lfloor \frac{m}{2} \rfloor$, $y_j > \lfloor \frac{n}{2} \rfloor$, and $z_j > \lfloor \frac{p}{2} \rfloor$,

- section 8: $x_j > \lfloor \frac{m}{2} \rfloor$, $y_j > \lfloor \frac{n}{2} \rfloor$, and $z_j > \lfloor \frac{p}{2} \rfloor$.

2.5.1 VH Algorithm in 3-Dimension Torus Network

A 3DT consists of eight 3DM problems. Apply the VH_{3DM} to each section.

Algorithm 2.7 (The VH_{3DT} Algorithm). *In a 3-D torus network $(m \times n \times p)$, given a source s(0, 0, 0) and a set of destinations $\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_i, y_i, z_i)\}$, where*

$0 \leq x_i \leq m-1$, $0 \leq y_i \leq n-1$, and $0 \leq z_i \leq p-1$. Multicasting requires a message to be sent from s to all (x_j, y_j, z_j)s, where $1 \leq j \leq i$.

1. *Assign the distances (D_j) for all destinations.*
   - *If the node is in section 1, $D_j = x_j + y_j + z_j$.*
   - *If the node is in section 2, $D_j = (m - x_j) + y_j + z_j$.*
   - *If the node is in section 3, $D_j = x_j + (n - y_j) + z_j$.*
   - *If the node is in section 4, $D_j = (m - x_j) + (n - y_j) + z_j$.*
   - *If the node is in section 5, $D_j = x_j + y_j + (p - z_j)$.*
   - *If the node is in section 6, $D_j = (m - x_j) + y_j + (p - z_j)$.*
   - *If the node is in section 7, $D_j = x_j + (n - y_j) + (p - z_j)$.*
   - *If the node is in section 8, $D_j = (m - x_j) + (n - y_j) + (p - z_j)$.*
2. *Construct the multicasting tree.*
   - *Start from the destination with maximum x value, and then maximum y value.*
   - *If the node is in section 1,*
     - *Follow the positive direction in the vertical dimension (z dimension),*
     - *Follow the positive direction in the vertical dimension (y dimension),*
     - *Follow the positive direction in the horizontal dimension (x dimension).*
   - *If the node is in section 2,*
     - *Follow the positive direction in the z dimension,*
     - *Follow the negative direction in the y dimension,*
     - *Follow the positive direction in the x dimension.*

- *If the node is in section 3,*
    - *Follow the negative direction in the z dimension,*
    - *Follow the positive direction in the y dimension,*
    - *Follow the positive direction in the x dimension.*
- *If the node is in section 4,*
    - *Follow the negative direction in the z dimension,*
    - *Follow the negative direction in the y dimension,*
    - *Follow the positive direction in the x dimension.*
- *If the node is in section 5,*
    - *Follow the positive direction in the z dimension,*
    - *Follow the positive direction in the y dimension,*
    - *Follow the negative direction in the x dimension.*
- *If the node is in section 6,*
    - *Follow the positive direction in the z dimension,*
    - *Follow the negative direction in the y dimension,*
    - *Follow the negative direction in the x dimension.*
- *If the node is in section 7,*
    - *Follow the negative direction in the z dimension,*
    - *Follow the positive direction in the y dimension,*
    - *Follow the negative direction in the x dimension.*
- *If the node is in section 8,*
    - *Follow the negative direction in the z dimension,*

- *Follow the negative direction in the y dimension,*
- *Follow the negative direction in the x dimension.*

3. *All the nodes in section 2-8 will eventually go back to source via the wraparound links.*

4. *Repeat the process for all destination nodes.*

5. *If more than one node has thesame x value, choose any one of them at random.*

The multicasting time satisfies,

$$D_{max} \leq TIME_{VH_{3DT}} \leq D_{max} + 5 \tag{2.13}$$

The total number of links depends on the pattern of destination nodes. The following procedure can be used to calculate the exact sum of total links in the VH_{3DT} algorithm.

Procedure 2.4. *The sum of total links.*

```
$\sum_{3DT} = 0;$
$X_{max1} = X_{max3} = X_{max5} = X_{max7} = 0;$
$X_{max2} = X_{max4} = X_{max6} = X_{max8} = m - 1;$
$Y_1 = Y_2 = Y_3 = Y_4 = Y_5 = Y_6 = Y_7 = Y_8 = 0;$
$Z_1 = Z_2 = Z_3 = Z_4 = Z_5 = Z_6 = Z_7 = Z_8 = 0;$
do
  take node $(x_j, y_j, z_J)$
  if node $(x_j, y_j, z_J)$ is in section 1,
      if $x_j > X_{max1}$
          $X_{max1} = x_j,$
      if $x_j$ is unique,
          then include $y_j$ and $z_p$ in the sum, $Y_1 = Y_1 + y_j, Z_1 = Z_1 + z_j$ .
```

```
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $Y_1 = Y_1 + y_j, Z_1 = Z_1 + z_j$ .
        otherwise
            include every $y_p$ and $z_p$ in the sum, $Y_1 = Y_1 + y_j, Z_1 = Z_1 + z_j$ .
else if node $(x_j, y_j, z_J)$ is in section 2,
    if $x_j < X_{max2}$
        $X_{max2} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ and $z_p$ in the sum, $Y_2 = Y_2 + y_j, Z_2 = Z_2 + z_j$ .
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $Y_2 = Y_2 + y_j, Z_2 = Z_2 + z_j$ .
        otherwise
            include every $y_p$ and $z_p$ in the sum, $Y_2 = Y_2 + y_j, Z_2 = Z_2 + z_j$ .
else if node $(x_j, y_j, z_J)$ is in section 3,
    if $x_j > X_{max3}$
        $X_{max3} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ and $z_p$ in the sum, $Y_3 = Y_3 + y_j, Z_3 = Z_3 + z_j$ .
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $Y_3 = Y_3 + y_j, Z_3 = Z_3 + z_j$ .
        otherwise
```

```
                include every $y_p$ and $z_p$ in the sum, $Y_3 = Y_3 + y_j, Z_3 = Z_3 + z_j$ .
else if node $(x_j, y_j, z_J)$ is in section 4,
    if $x_j < X_{max4}$
        $X_{max4} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ and $z_p$ in the sum, $Y_4 = Y_4 + y_j, Z_4 = Z_4 + z_j$ .
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $Y_4 = Y_4 + y_j, Z_4 = Z_4 + z_j$ .
        otherwise
            include every $y_p$ and $z_p$ in the sum, $Y_4 = Y_4 + y_j, Z_4 = Z_4 + z_j$ .
else if node $(x_j, y_j, z_J)$ is in section 5,
    if $x_j > X_{max5}$
        $X_{max5} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ and $z_p$ in the sum, $Y_5 = Y_5 + y_j, Z_5 = Z_5 + z_j$ .
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $Y_5 = Y_5 + y_j, Z_5 = Z_5 + z_j$ .
        otherwise
            include every $y_p$ and $z_p$ in the sum, $Y_1 = Y_1 + y_j, Z_1 = Z_1 + z_j$ .
else if node $(x_j, y_j, z_J)$ is in section 6,
    if $x_j < X_{max6}$
        $X_{max6} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ and $z_p$ in the sum, $Y_6 = Y_6 + y_j, Z_6 = Z_6 + z_j$ .
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $Y_6 = Y_6 + y_j, Z_6 = Z_6 + z_j$ .
        otherwise
            include every $y_p$ and $z_p$ in the sum, $Y_6 = Y_6 + y_j, Z_6 = Z_6 + z_j$ .
else if node $(x_j, y_j, z_J)$ is in section 7,
    if $x_j > X_{max7}$
        $X_{max7} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ and $z_p$ in the sum, $Y_7 = Y_7 + y_j, Z_7 = Z_7 + z_j$ .
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
            then include only $y_p$ and $z_p$ in the sum, $Y_7 = Y_7 + y_j, Z_7 = Z_7 + z_j$ .
        otherwise
            include every $y_p$ and $z_p$ in the sum, $Y_7 = Y_7 + y_j, Z_7 = Z_7 + z_j$ .
else if node $(x_j, y_j, z_J)$ is in section 8,
    if $x_j < X_{max8}$
        $X_{max8} = x_j$,
    if $x_j$ is unique,
        then include $y_j$ and $z_p$ in the sum, $Y_8 = Y_8 + y_j, Z_8 = Z_8 + z_j$ .
    else if $x_j = x_{j+1} = \cdots = x_p$
        if $y_j = y_{j+1} = \cdots = y_p$
        then include only $y_p$ and $z_p$ in the sum, $Y_8 = Y_8 + y_j, Z_8 = Z_8 + z_j$ .
    otherwise
        include every $y_p$ and $z_p$ in the sum, $Y_8 = Y_8 + y_j, Z_8 = Z_8 + z_j$ .
enddo
```

$$\sum_{3DT} = \sum \begin{cases} X_{max1} + Y_1 + Z_1, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 1;} \\ m - X_{max2} + Y_2 + Z_2, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 2;} \\ X_{max3} + 1 + Y_3 + Z_3, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 3;} \\ m - X_{max4} + 1 + Y_4 + Z_4, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 4;} \\ X_{max5} + Y_5 + Z_5, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 5;} \\ m - X_{max6} + Y_6 + Z_6, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 6;} \\ X_{max7} + 1 + Y_7 + Z_7, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 7;} \\ m - X_{max8} + 1 + Y_8 + Z_8, & \text{if any node } (x_j, y_j, z_j) \text{ is in section 8.} \end{cases}$$

output $\sum_{3DT}$.

The total number of links can be described as

$$LINK_{VH_{3DT}} = \sum . \tag{2.14}$$

where $\sum$ *is the output of Procedure 2.4.*

Example 2.12. *In 3-dimensional torus network* $(10 \times 9 \times 4)$, *the message is sent from node* $s(0,0,0)$ *to a set of nodes* $\{(2,1,0),(4,3,0),(3,4,0),(8,7,0),(9,0,3),(9,7,2)\}$. *By using the* VH_{3DT} *Algorithm, node* $(9,0,3)$, *which is in section 6, has the minimum distance value of 2. Nodes* $(4,3,0)$ *and* $(3,4,0)$, *which are in section 1, have the maximum distance value* $D_{max} = 7$. *First find the path for node* $(4,3,0)$, *which follows the order of z dimension, y*

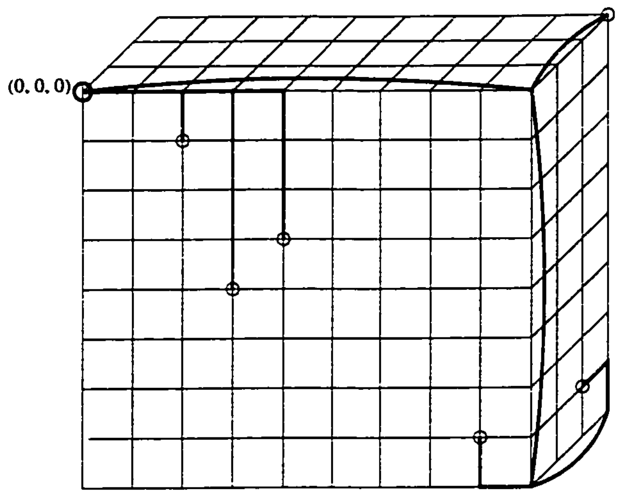


Figure 2.17: Multicast routing in the VH_{3DT} algorithm

dimension and x dimension, then find the path for node $(3,4,0)$*, followed by nodes* $(9,7,2)$, $(8,7,0)$, $(2,1,0)$, *and* $(9,0,3)$. *The final routing is shown in Figure 2.17.*

The multicasting time is $TIME_{VH_{3DT}} = D_{max} + 1 = 8$,

The number of links is $LINK_{VH} = 20$.

2.5.2 *DIST* Algorithm in 3-Dimensional Torus Network

The *DIST* algorithm can also be applied to the 3-dimensional torus network.

Algorithm 2.8 (The $DIST_{3DT}$ Algorithm). *In a 3D torus network* $(m \times n \times p)$*, given a source s(0, 0, 0) and a set of destinations* $\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_i, y_i, z_i)\}$*, where* $0 \le x_i \le m-1$, $0 \le y_i \le n-1$, *and* $0 \le z_i \le p-1$. *Multicasting requires a message to be sent from s to all* $(x_j, y_j, z_j)s$*, where* $1 \le j \le i$.

1. *Assign the distances* (D_j) *for all destinations.*

   - *If the node is in section 1,* $D_j = x_j + y_j + z_j$.

   - *If the node is in section 2,* $D_j = (m - x_j) + y_j + z_j$.

- *If the node is in section 3,* $D_j = x_j + (n - y_j) + z_j$.
- *If the node is in section 4,* $D_j = (m - x_j) + (n - y_j) + z_j$.
- *If the node is in section 5,* $D_j = x_j + y_j + (p - z_j)$.
- *If the node is in section 6,* $D_j = (m - x_j) + y_j + (p - z_j)$.
- *If the node is in section 7,* $D_j = x_j + (n - y_j) + (p - z_j)$.
- *If the node is in section 8,* $D_j = (m - x_j) + (n - y_j) + (p - z_j)$.

2. *Build the routes so that all the destinations connect to the source: start from the destination with the minimal distance, find the shortest path to the existing multicasting tree in its own section.*
   - *Start from the destination with the minimal distance.*
   - *Find the shortest path to the existing multicasting tree in its own section.*
   - *Repeat the procedure until every destination node is included in the multicasting tree.*

3. *If more than one node has the same distance in a section, alternatively take* (x_j, y_j, z_j) *for which* x_j *is the minimum possible value and take* (x_k, y_k, z_j) *for which* x_k *is the maximum possible value in its section.*
   - *If these nodes have the same* x *value, compare the* y *value and follow the same order.*

The multicasting time in the $DIST_{3DT}$ algorithm satisfies,

$$D_{max} \leq TIME_{DIST_{3DT}} \leq min\{m + n + \frac{p}{2}, 5D_{max}\}. \tag{2.15}$$

The total number of links depends on the pattern of destination nodes.

$LINK_{DIST_{3DT}} \leq$

$$\sum \begin{cases} x_{max1} + \sum y_{j1} + \sum z_{j1}, & \text{for } (x_{j1}, y_{j1}) \text{ in section 1;} \\ m - x_{max2} + \sum y_{j2} + \sum z_{j2}, & \text{for } (x_{j2}, y_{j2}) \text{ in section 2;} \\ x_{max3} + 1 + \sum (n - y_{j3} - 1) + \sum z_{j3}, & \text{for } (x_{j3}, y_{j3}) \text{ in section 3;} \\ m - x_{max4} + 1 + \sum (n - y_{j4} - 1) + \sum z_{j4}, & \text{for } (x_{j4}, y_{j4}) \text{ in section 4;} \\ x_{max5} + \sum y_{j5} + \sum (p - z_{j5} - 1), & \text{for } (x_{j5}, y_{j5}) \text{ in section 5;} \\ m - x_{max6} + \sum y_{j6} + \sum (p - z_{j6} - 1), & \text{for } (x_{j6}, y_{j6}) \text{ in section 6;} \\ x_{max7} + 1 + \sum (n - y_{j7} - 1) + \sum (p - z_{j7} - 1), & \text{for } (x_{j7}, y_{j7}) \text{ in section 7;} \\ m - x_{max8} + 1 + \sum (n - y_{j8} - 1) + \sum (p - z_{j8} - 1), & \text{for } (x_{j8}, y_{j8}) \text{ in section 8.} \end{cases} \tag{2.16}$$

The equations 2.11 and 2.15 are identical, but the value of D_{max} in equation 2.15 is much smaller than that from equation 2.11, since the diameter of the torus is smaller than the diameter of the mesh network by a factor of 2.

Consider the same network as in example 2.12 to illustrate the $DIST_{3DT}$ Algorithm.

Example 2.13. *In the 3-dimensional torus network* $(10 \times 9 \times 4)$, *the message is sent from node* $s(0,0,0)$ *to a set of nodes* $\{(2,1,0),(4,3,0),(3,4,0),(8,7,0),(9,0,3),(9,7,2)\}$. *By using the* $DIST_{3DT}$ *Algorithm, node* $(9,0,3)$, *which is in section 6, has the minimum distance value of 2. Nodes* $(4,3,0)$ *and* $(3,4,0)$, *which are in section 1, have the maximum distance value* $D_{max} = 7$. *First find the shortest path for node* $(9,0,3)$, *which uses only two wraparound links* $(0,0,0) \rightarrow (9,0,0) \rightarrow (9,0,3)$. *Next find the shortest path for node* $(2,1,0)$, *which is* $(0,0,0) \rightarrow (2,0,0) \rightarrow (2,1,0)$. *Similarly, find paths for every destination. The final routing is shown in Figure 2.18.*

The multicasting time is $TIME_{VH_{3DT}} = D_{max} + 1 = 8$.

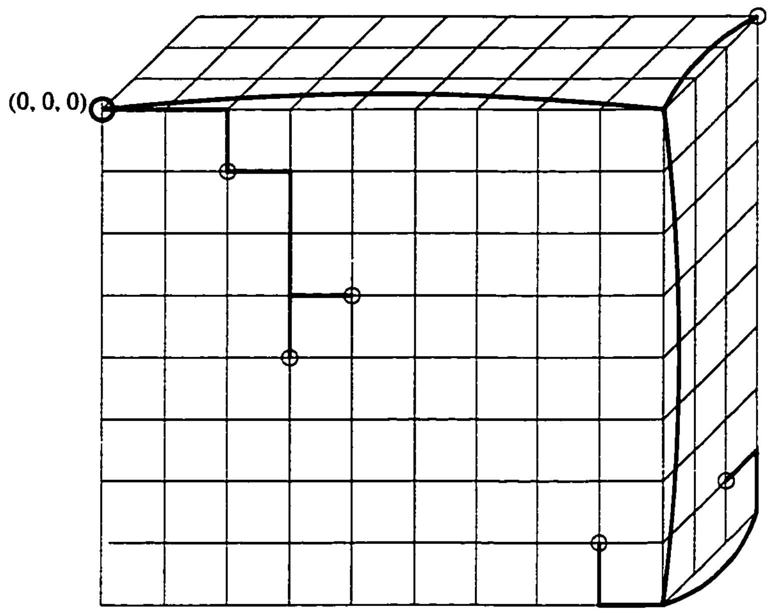


Figure 2.18: Multicast routing in the $DIST_{3DT}$ algorithm

The number of links is $LINK_{VH} = 16$.

Compare the results with those from example 2.12.

$TIME_{VH_{3DT}} = TIME_{DIST_{3DT}}$ *and* $LINK_{VH_{3DT}} > LINK_{DIST_{3DT}}$.

2.5.3 Comparisons of the VH_{3DT} and the $DIST_{3DT}$ Algorithms

The 3-dimensional torus consists of eight 3DM subgraphs. Therefore, the 3DT network has similar properties to those of the 3DM network.

Proposition 2.10. *In the 3-dimensional torus network, the multicasting time in the* VH_{3DT} *algorithm is equal to or less than the multicasting time in the* $DIST_{3DT}$ *algorithm.*

$$TIME_{VH_{3DT}} \leq TIME_{DIST_{3DT}}$$

Proposition 2.11. *In the 3-dimensional torus network, the number of links in the* VH_{3DT} *algorithm is equal to or greater than the number of links in the* $DIST_{3DT}$ *algorithm.*

$$LINK_{VH_{3DT}} \geq LINK_{DIST_{3DT}}$$

In the 3-dimensional torus network, the VH_{3DT} algorithm uses less multicasting time, but generates more traffic as compared to the $DIST_{3DT}$ algorithm.

2.6 n-Dimensional Torus/Mesh Network

All the algorithms developed so far can be used with n-dimensional mesh and torus networks. The n-dimensional mesh network has the same properties as those from 2- and 3-dimensional mesh network. When applying the dimensional ordered routing algorithm (VH algorithm) to n-dimensional mesh, the message is first routed in the highest dimension, and routing then proceeds to each dimension, in descending order, until the routing path reaches the source node. Routing in a particular dimension is always complete before routing in the next dimension begins. The dimension ordered routing algorithm is simple and easy to implement, but it may generate more traffic than the shortest path approach. The Distance routing algorithm ($DIST$ algorithm) can reduce the total traffic, which enable finding the shortest path to the existing multicasting tree for each destination. However, it may take more multicasting time, and be more complex to implement, especially for the n-dimensional mesh.

The n-dimensional torus is identical to the n-dimensional mesh, except that the torus network has additional wraparound links in every dimension. All algorithms for mesh are also suitable for the torus network. The only difference is that the torus network use less multicasting time and traffic than the mesh network, since the diameter of the torus is smaller by a factor of two than the mesh network.

In the n-dimensional mesh or torus network, the multicasting time in the VH algorithm is equal to or less than the multicasting time in the $DIST$ algorithm. On the other hand, the number of links in the VH algorithm is equal to or greater than the number of links in the $DIST$ algorithm.

$$TIME_{VH} \leq TIME_{DIST}$$

$$LINK_{VH} \geq LINK_{DIST}$$

Chapter 3

Implementation and analysis

3.1 System Requirements

This chapter deals with the problem of translating design models into an implementation for the multicasting communication system. Based on the design model representation discussed in the previous chapter, I will restate this problem as that of transmitting a message from the source node to a set of destination nodes. The solution entails constructing a communication network to model the application and implementation for the multicasting problem.

The implementation is based in both the VH and $DIST$ algorithms developed for 2DM, 2DT, 3DM, and 3DT networks.

3.2 Software Technology Used

There are many different approaches to the software development. Object-oriented technologies do lead to a number of inherent benefits that provide advantages at the technical level. Object-oriented systems are easier to maintain because the objects are independent.

They may be understood and modified as entities which work independantly of one another. Changing the implementation of an object or adding services will not affect other system's objects. Due to the advantage of object-oriented systems, an object-oriented strategy is used throughout the process in the development of the multicasting communication system. Object-oriented analysis (OOA) is used in requirement analysis and object-oriented design (OOD) is used in design phases.

There are three different mechanisms in OOD, which are *inheritance*, *composition*, and *aggregation*. The *inheritance* mechanism supports class hierarchies (the *"is-a"* relation). On the other hand, the *composition* mechanism is a concept that leads to aggregate objects, and the *aggregation* mechanism relation (the *"has-a"* relation) supports the part-whole concept. *Inheritance* supports the generalization-specialization relation, whereas *aggregation* is useful in depicting relations involving containment and sharing.

3.3 System Implementation

In the implementation, C++, as OO programming language, is used to realize the design and implement the multicasting communication model, because the object-oriented programming (OOP) languages make an object-oriented design easier to implement.

The mesh and the torus networks have similar properties, both of them contain nodes in their netowks. Create the *"node"* class to store the node's information such as its coordinator, its distance cost from the originator, and its previous node in the routing, etc.

3.3.1 Implementation in 2-Dimensional Mesh Network

We implemented the VH and $DIST$ algorithms and evaluated their performance in the metrics of *time* and *traffic* respectively. Clearly, the *average additional traffic*, defined as

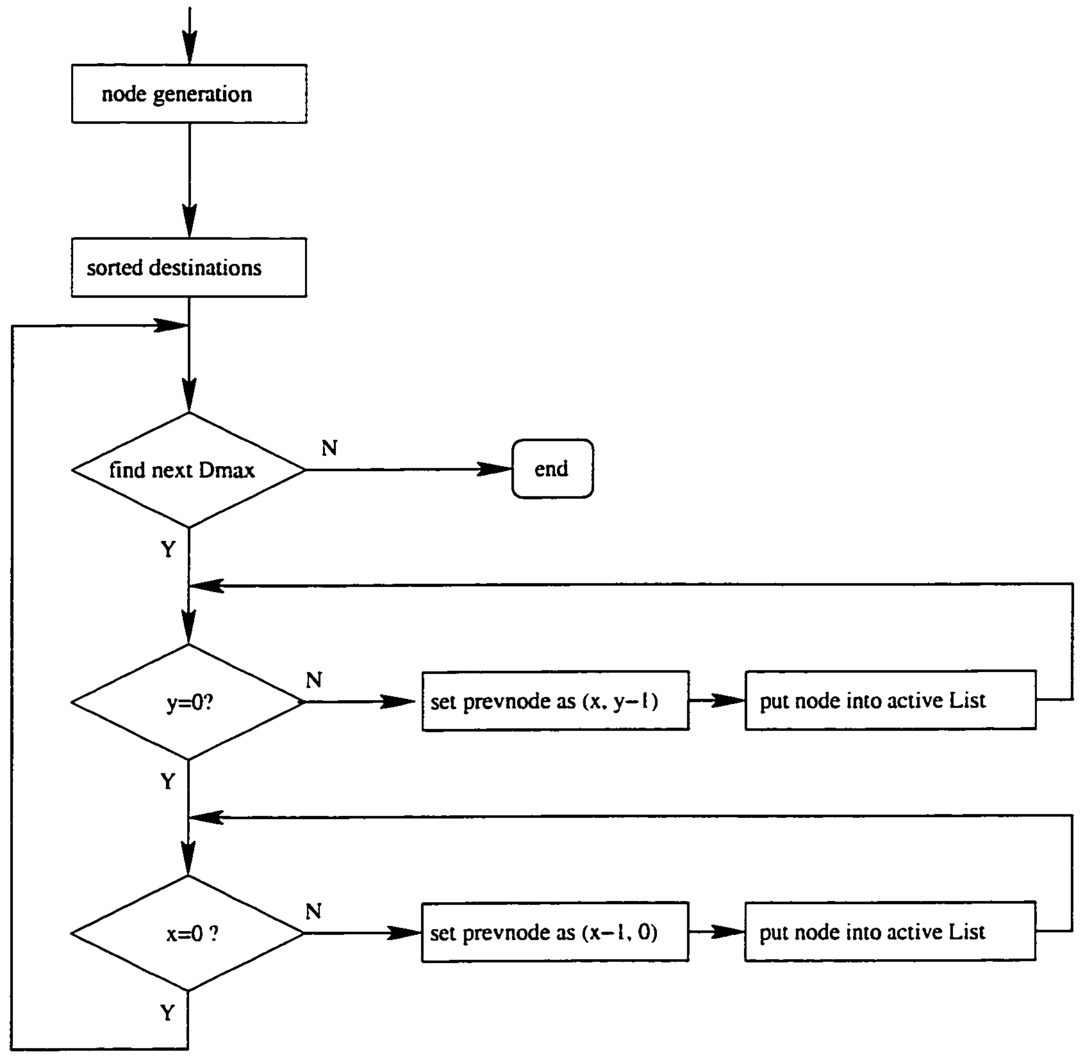


Figure 3.1: Data flow diagram for the VH_{2DM} Algorithm

the average amount of total traffic minus the number of destination nodes, is a reasonable measure for the amount of traffic for the multicasting algorithm. [21]

VH Algorithm in the 2DM Network

In the VH_{2DM} algorithm, the multicasting path follows the vertical dimension (y value) first until $y = 0$, and then turns to the horizontal dimension (x value) going back to the source node $(0, 0)$. The data flow is described in Figure 3.1.

We study the performance of the proposed VH_{2DM} algorithm for a 20×20 2DM. A large number of randomly generated multicasting sets with a different number of destinations are tested to measure the time and average additional traffic generated by these algorithms. The number of destination nodes, k, is chosen from 50, 100, and up to 350. For a given k, a random number generator generates k nodes between the range of $(0,0)$ and $(19,19)$, which represents k destinations. The results are shown in Table 3.1.

$DIST$ Algorithm in 2DM Network

In the $DIST_{2DM}$ algorithm, a *List* class is used to store the active nodes included in the multicasting tree. Initially, only the source node is in the *List* class. Start from the node with minimum distance, find the shortest path to the active List, and set the previous node for all nodes in the path. Meanwhile put every node within the shortest path into the active List. Repeat the process until the paths are found for all destination nodes. The data flow is described in Figure 3.2.

We study the performance of the $DIST_{2DM}$ algorithm in a 20×20 2DM, and the same set of constraint for the $DIST_{2DM}$ algorithm is used. Many randomly generated multicasting sets with a different number of destinations are tested in order to measure the time and average additional traffic generated by the algorithm. The number of destination nodes, k, is chosen from 50, 100, up to 350. For a given k, a random number generator generates k nodes between the range of $(0,0)$ and $(19,19)$, which represent k destinations.The results are shown in Table 3.1.

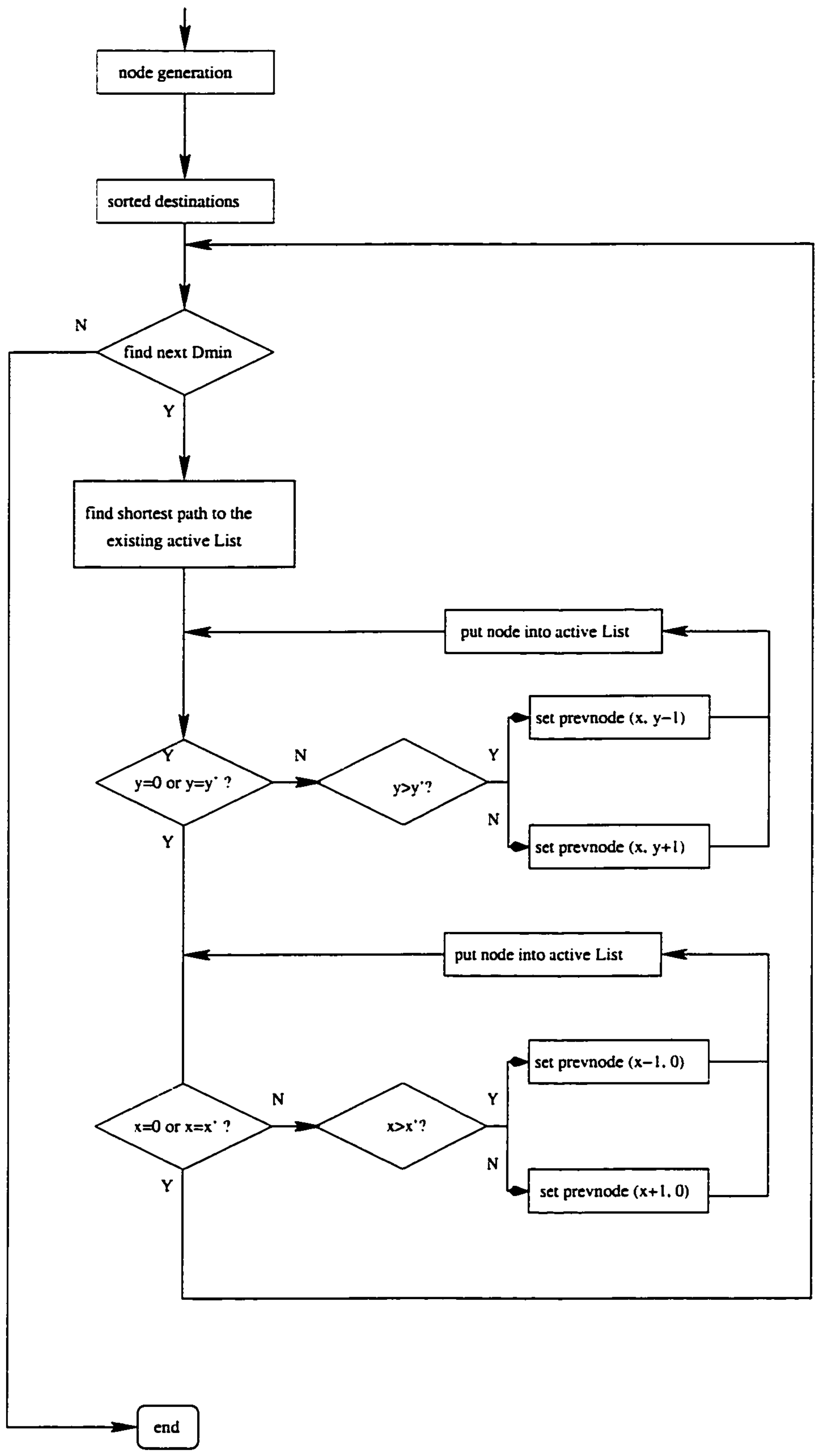


Figure 3.2: Data flow diagram for the $DIST_{2DM}$ Algorithm

	Multicasting Time		Average Additional Traffic	
nodes	VH_{2DM}	$DIST_{2DM}$	VH_{2DM}	$DIST_{2DM}$
50	35.45	37.3	211.95	71.25
100	36.8	37.25	234.95	64.55
150	37.1	37.95	207.7	46.65
200	37.1	37.95	167.8	22.7
250	37.15	37.4	125.2	14.25
300	37.7	37.75	82.45	6.175
350	37.5	37.8	33.5	2.15

Table 3.1: Results in a 20 × 20 2DM network

Analysis of 2-dimensional mesh

According to Table 3.1, the VH_{2DM} algorithm spends less multicasting time than the $DIST_{2DM}$ algorithm. However, the $DIST_{2DM}$ algorithm reduces a great amount of traffic as compared with the VH_{2DM} algorithm. In addition, the gap between the $TIME_{VH_{2DM}}$ and the $TIME_{DIST_{2DM}}$ narrows when the number of destinations k grows. The comparison of the multicasting time is shown in Figure 3.3. Figure 3.4 illustrates the comparison of the average additional traffic.

3.3.2 Simulation in the 2-Dimensional Torus Network

VH Algorithm in the 2DT Network

Consider a 20 × 20 2DT network, and assume initially that the message is at the originator and a set of destination nodes is randomly generated within the network. In the VH_{2DT} algorithm, first decide the section number for every destination node, after which each node

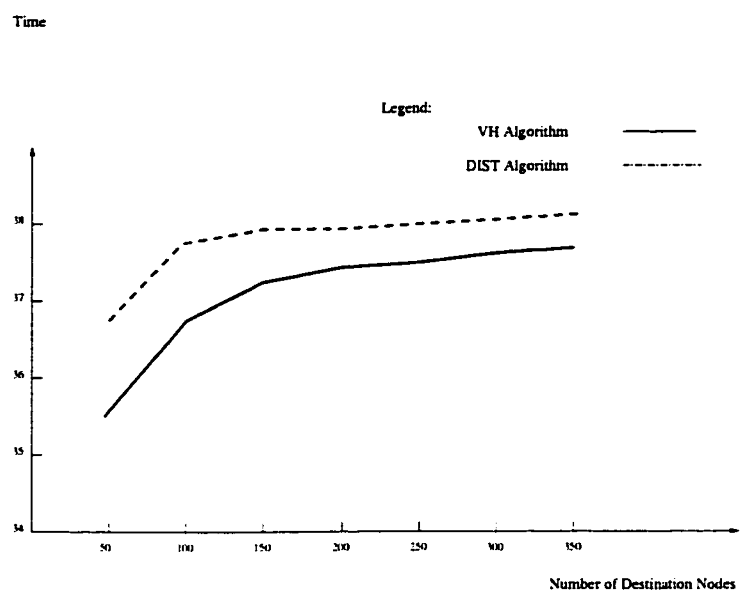


Figure 3.3: The multicasting time in a 20 × 20 2DM network

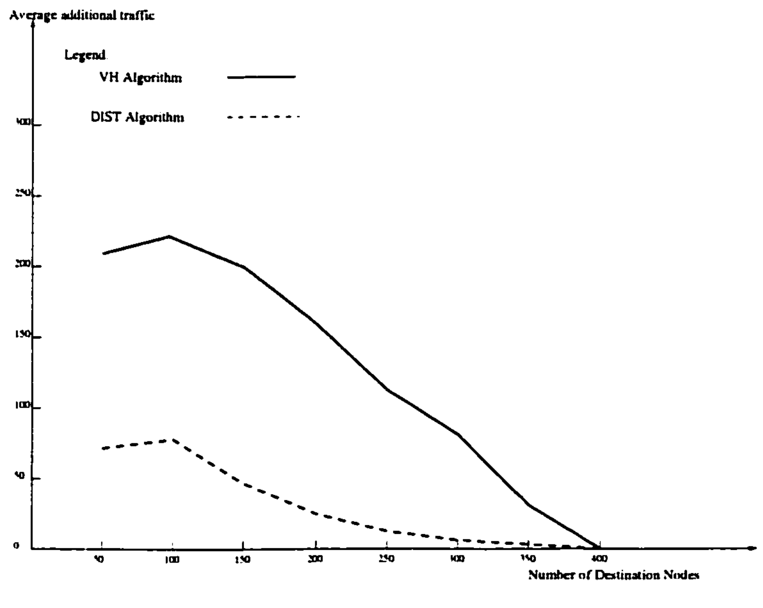


Figure 3.4: The average additional traffic in a 20 × 20 2DM network

performs the routing strategy in its own section, except when using the wraparound links.

- If the node (x_j, y_j) is in section 1, its routing follows the vertical dimension towards the node $(x_j, 0)$, then turns to the horizontal dimension until it reaches the source node $(0, 0)$;
- If the node (x_j, y_j) is in section 2, its routing follows the vertical dimension towards the node $(x_j, 0)$, then turns to the horizontal dimension and towards the node $(19, 0)$. Finally, it uses the wraparound link to get back to the source $(0, 0)$.
- If the node (x_j, y_j) is in section 3, its routing follows the vertical dimension towards node $(x_j, 19)$, then turns to the horizontal dimension and towards the node $(0, 19)$. Finally, it uses the wraparound link to get back to the source $(0, 0)$.
- If the node (x_j, y_j) is in section 4, its routing follows the vertical dimension to node $(x_j, 19)$, then turns to the horizontal dimension and towards the node $(19, 19)$. Finally, it uses the wraparound links to get back to the source $(0, 0)$.

We study the performance of the VH_{2DT} algorithm in a 20×20 2DT network. A large number of randomly generated multicasting sets with a different number of destinations are tested to measure the time and traffic generated by the algorithm. The number of destination nodes, k, is chosen from 50, 100, and up to 350. For a given k, a random number generator generates k nodes within the range between $(0, 0)$ and $(19, 19)$, which represent k destinations. The results are shown in Table 3.2.

DIST Algorithm in the 2DT Network

In the $DIST_{2DT}$ algorithm, we use an active *List* class to store the node included in the multicasting tree. Unlike the $2DM$ network, the destination node needs to perform the

	Multicasting Time		**Average Additional Traffic**	
nodes	VH_{2DT}	$DIST_{2DT}$	VH_{2DT}	$DIST_{2DT}$
50	18.4	18.75	146.85	91.65
100	18.9	19.9	182.8	94.15
150	19.45	20.1	175.55	79.85
200	19.2	20.2	149.55	62.15
250	19.8	20.2	114.3	40.35
300	19.3	19.8	74.1	17.45
350	19.6	19.7	32.45	8.95

Table 3.2: Results in a 20×20 2DT network

routing strategy within its own section. Initially, only the source node is in the *List* class. Start from the node with a minimum distance, find the shortest path to the active List in its section, and set the path. Meanwhile put every node within the shortest path onto the active List. Repeat the process until it finds the shortest paths for all of the destination nodes.

To illustrate the $DIST_{2DT}$ algorithm, consider the 20×20 2DT. A large number of randomly generated multicasting sets with a different number of destinations are tested to measure the time and traffic generated by the algorithm. The number of destination nodes, k, is chosen from 50, 100, and up to 350, based on the average of ten times at each spot. For a given k, a random number generator generates k nodes between the range of $(0, 0)$ and $(19, 19)$, which represent k destinations. The results are shown in Table 3.2.

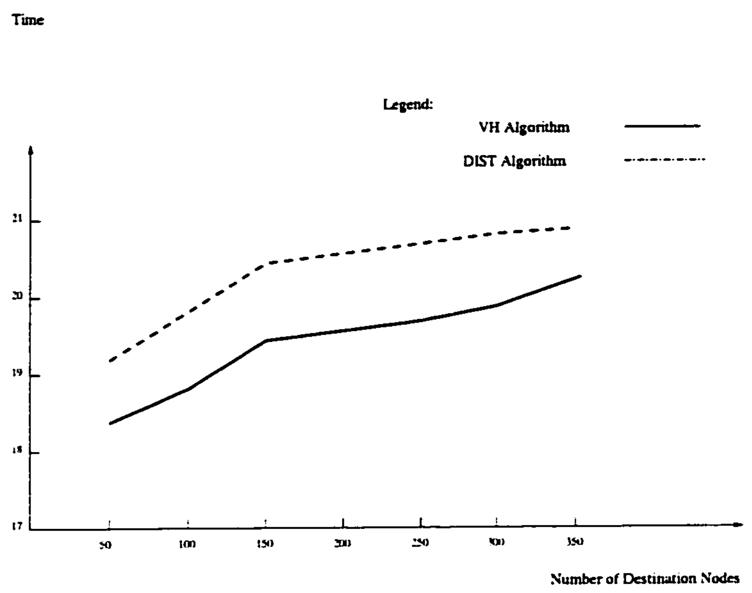


Figure 3.5: The multicasting time in a 20 × 20 2DT network

Analysis of 2-dimensional torus

According to Table 3.2, the VH_{2DT} algorithm spends less multicasting time than the $DIST_{2DT}$ algorithm (Figure 3.5). However, the $DIST_{2DT}$ algorithm can reduce a great amount of traffic compared to the VH_{2DT} algorithm (Figure 3.6). The relationship is similar to that of the $2DM$ network, but the multicasting time and traffic in the 2-D torus are much smaller than those from the 2-D mesh.

3.3.3 Simulation in the 3-Dimension Mesh Network

We assume initially that the message is at the originator, and a set of destination nodes is randomly generated within the network. The data flow for the VH_{3DM} algorithm is described in Figure 3.7 and the data flow for the $DIST_{3DM}$ algorithm is described in Figure 3.8.

To illustrate the performance of the proposed 3DM algorithms, a 20 × 20 × 20 3DM was

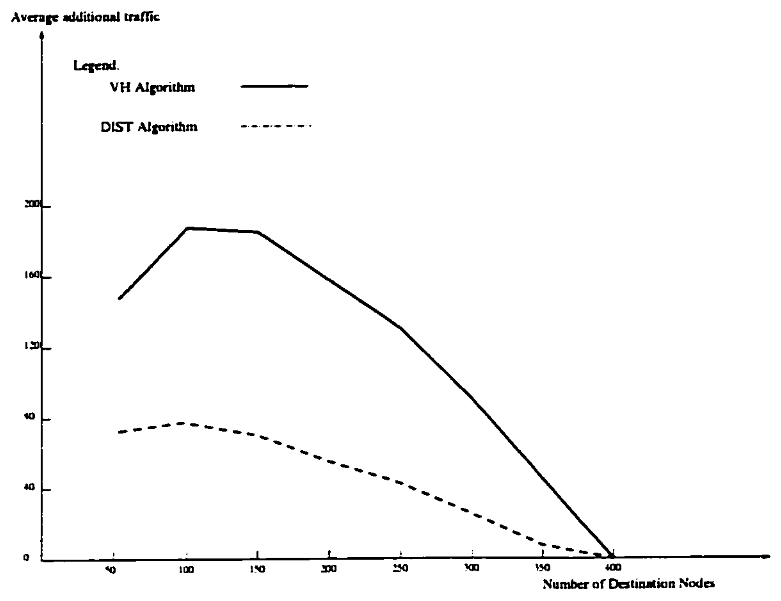


Figure 3.6: The average additional traffic in a 20 × 20 2DT network

considered. A large number of randomly generated multicasting sets with a different number of destinations are tested to measure the time and traffic generated by the algorithm. The number of destination nodes, k, is chosen from 100, 200, and up to 1000. For a given k, a random number generator generates k nodes between the range of $(0, 0, 0)$ and $(19, 19, 19)$, which represents k destinations.

The VH_{3DM} algorithm spends less multicasting time than the $DIST_{3DM}$ algorithm (Figure 3.9). However, the $DIST_{3DM}$ algorithm reduces a great amount of traffic compared to the VH_{3DM} algorithm (Figure 3.10).

3.3.4 Simulation in 3-Dimension Torus Network

In a 3DT network, we first decided which section the destination node was in, and then performed the routing strategy in its own section. The routing can not cross different sections except by using the wraparound links:

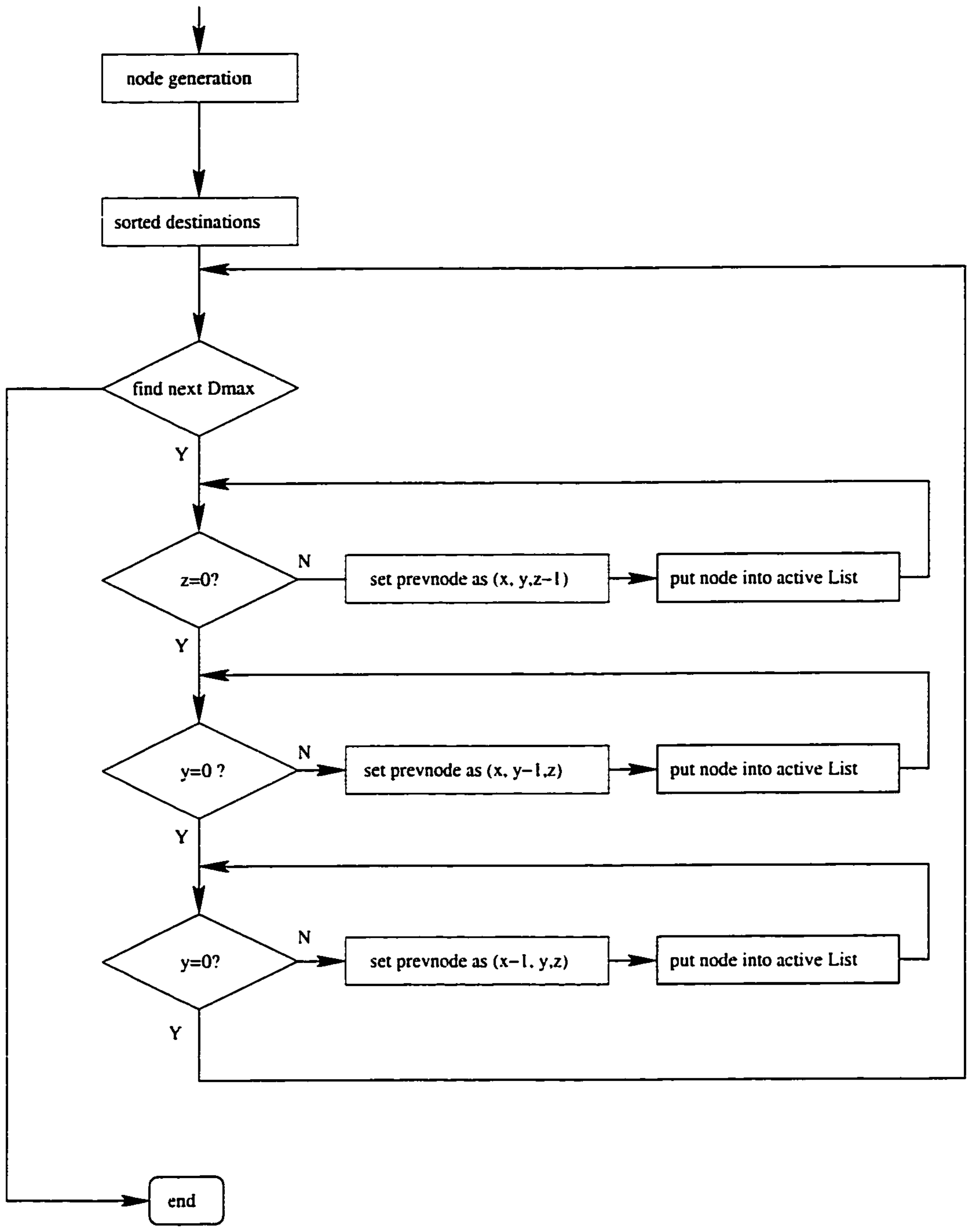

node generation
sorted destinations
find next Dmax
Y
z=0?
N
set prevnode as (x, y,z−1)
put node into active List
Y
y=0 ?
N
set prevnode as (x, y−1,z)
put node into active List
Y
y=0?
N
set prevnode as (x−1, y,z)
put node into active List
Y
end

Figure 3.7: Data flow diagram for the VH_{3DM} Algorithm

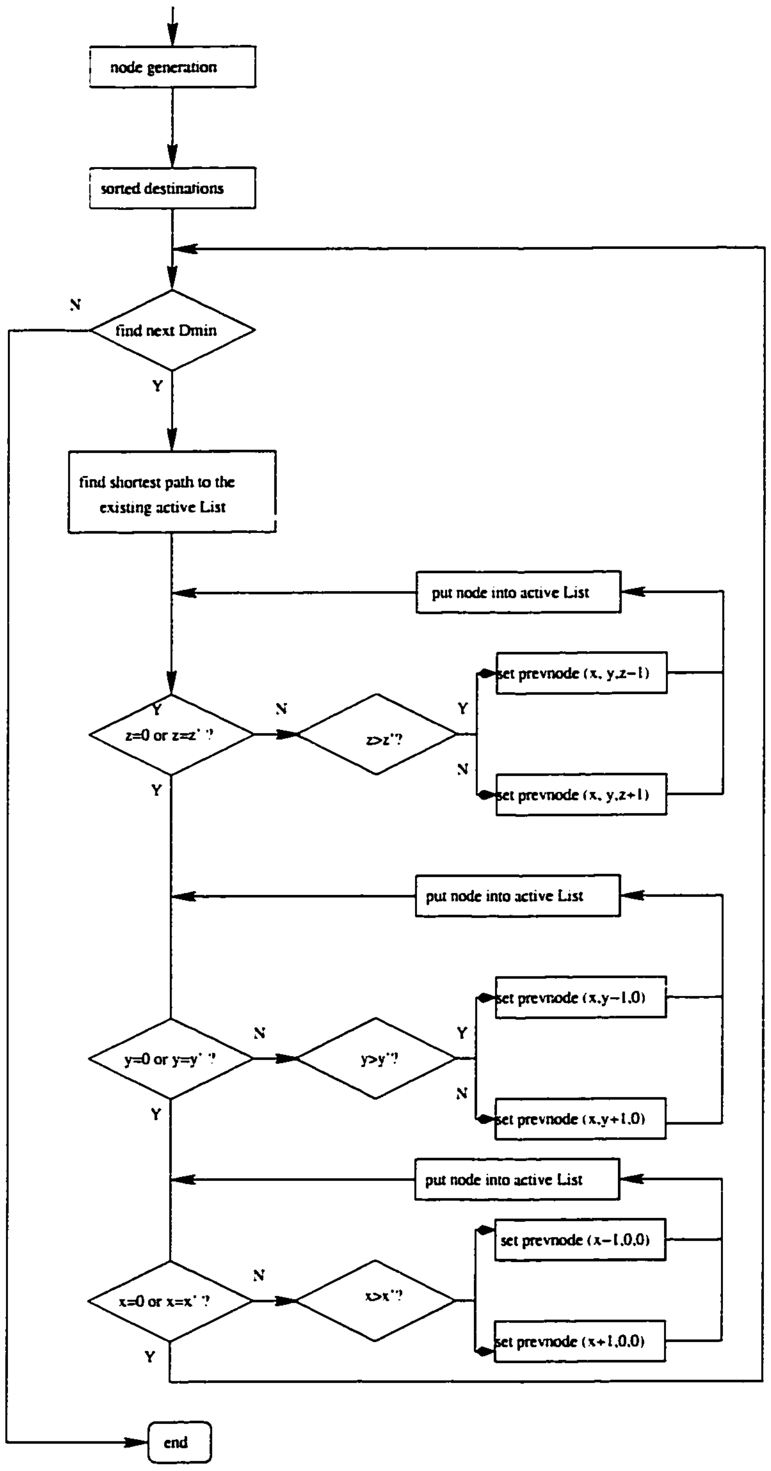


Figure 3.8: Data flow diagram for the $DIST_{3DM}$ Algorithm

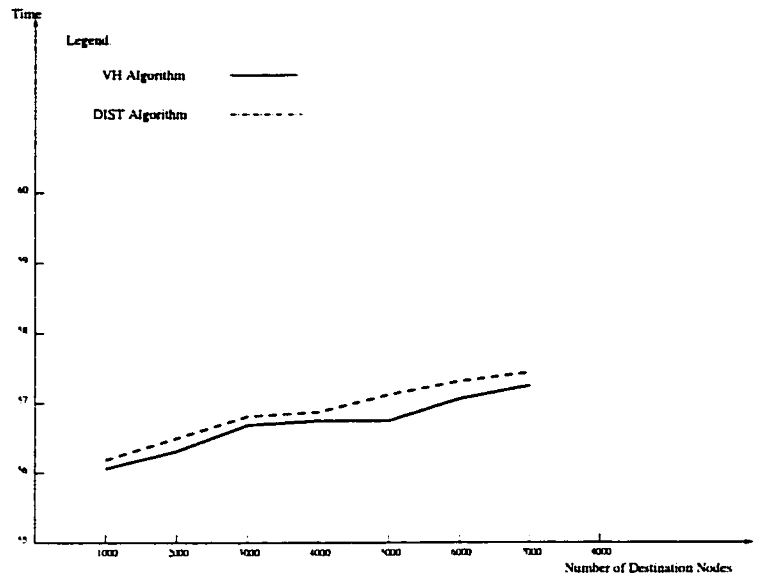


Figure 3.9: The multicasting time in a $20 \times 20 \times 20$ 3DM network

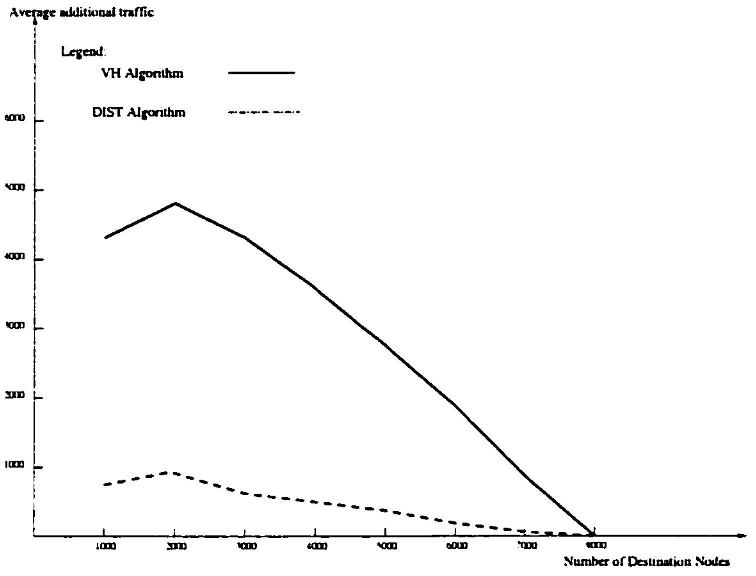


Figure 3.10: The average additional traffic in a $20 \times 20 \times 20$ 3DM network

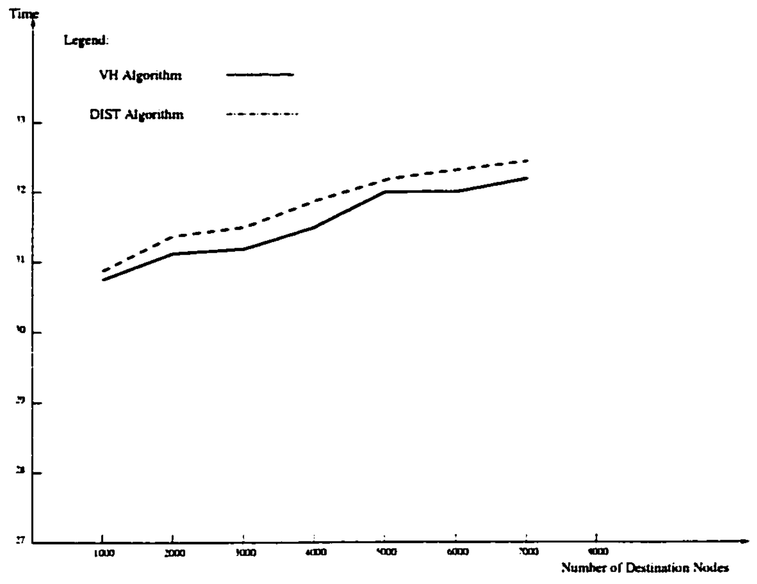


Figure 3.11: The multicasting time in a 20 × 20 × 20 3DT network

The program was run in the 20 × 20 × 20 3DT by using 100, 200, and up to 1000 destination nodes, as well as running the program ten times for each set of destination nodes, where all destination nodes are generated randomly. The multicasting time is illustrated in Figure 3.11, and the comparisons of average additional traffic is displayed in Figure 3.12.

3.4 Comparing the *VH* and the *DIST* algorithms with the heuristic algorithms for hypercube

The hypercube network is one of the most important interconnection structures. The algorithms developed in this thesis can also be applied to the hypercube network. In an n-dimensional torus ($m \times \cdots \times m$), when $m = 2$, it becomes an n-dimensional hypercube. The paths of the VH algorithm are the shortest paths between the source and destination nodes and the multicasting time is minimal. Applying the VH algorithm to the n-cube,

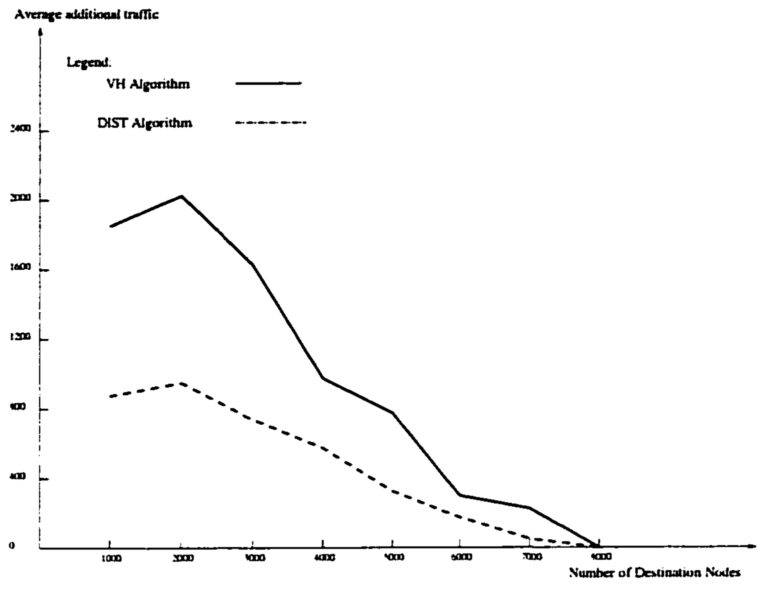


Figure 3.12: The average additional traffic in a 20 × 20 × 20 3DT network

the multicasting time is optimal, which is the same as the heuristic algorithms for the hypercube discussed in Chapter 1. However, the average additional traffic generated by the VH algorithm is similar to the SS's and SEY's, but it is much smaller than that of LEN's.

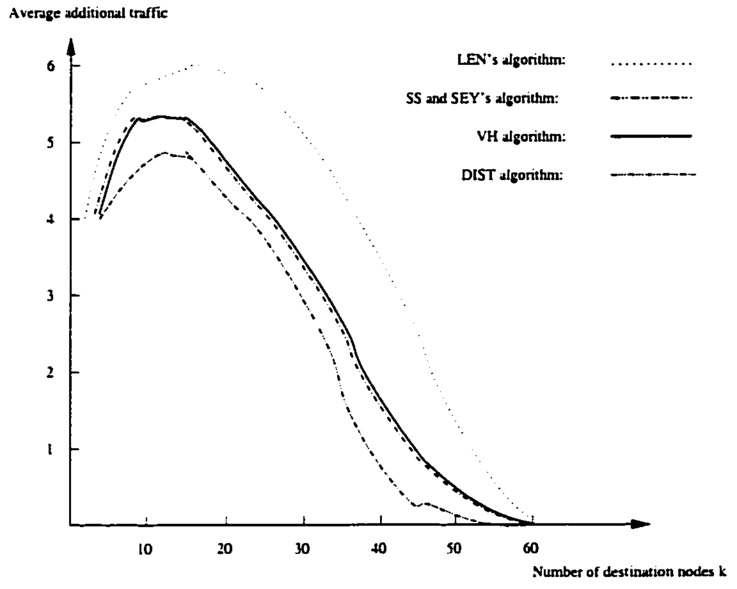


Figure 3.13: The average additional traffic in a 6-cube

In addition, the VH algorithm has the time complexity of $O(kn)$ for the n-cube, which is much less than the complexities of each of the three algorithms for the hypercube, compared to the time complexities of LEN's $(O(nk + n^2))$, SS's $(O(nN))$, and SEY's $(O(nN))$. When applying the $DIST$ algorithm to the hypercube, it can reduce the average additional traffic further. However, the multicasting time in the $DIST$ algorithm is greater than those of other algorithms. The comparison of average additional traffic in a 6-cube is shown in Figure 3.13.

Chapter 4

Algorithm Complexity

4.1 Complexity of 2DM Network

The Running time of VH_{2DM} Algorithm

In the 2DM network ($m \times n$), assume $N = m \times n$ is the total number of nodes, k is the number of destination nodes in the multicasting communication network, and the maximum distance is $D = m + n$. The worst case running time of each step is analyzed as follows:

- destination generation: $O(k)$,

- finding the path: $O(k(\sum_{i=1}^{n}(y_i) + \sum_{i=1}^{m} x_i)) = O(kD)$,

Hence the overall worst case running time of the VH_{2DM} algorithm is $O(kD)$.

The Running time of $DIST_{2DM}$ Algorithm

The worst case running time of each step in the $DIST_{2DM}$ Algorithm is analyzed as follows:

- destination generation: $O(k)$,

- finding the shortest path: $O(Dk(N + \sum y_i + \sum x_i)) = O(DkN)$,

Hence the overall worst-case running time of the $DIST_{2DM}$ algorithm is $O(DkN)$.

If the 2DM has the same number of nodes in each dimension, say $m = n$, then the total number of nodes is $N = m \times m = m^2$, and the maximum distance is $D = m+m = 2m$. Thus the complexity of the VH_{2DM} is $O(kD) = O(k2m) = O(km)$, and the complexity of the $DIST_{2DM}$ is $O(DkN) = O(2mkm^2) = O(km^3)$. Clearly, the complexity of the $DIST_{2DM}$ algorithm is much greater than that of the VH_{2DM} algorithm.

4.2 Complexity of 2DT Network

The Running time of VH_{2DT} Algorithm

In the 2DT network ($m \times n$), assume $N = m \times n$ is the total number of nodes in the 2-dimensional torus network, k is the number of destination nodes in the multicasting communication network, and the maximum distance is $D = \frac{m+n}{2}$. The (worst case) running time of each step is analyzed as follows:

- destination generation: $O(k)$,

- finding the path: $O(k(\sum_{i=1}^{n/2} y_i + \sum_{i=1}^{m/2} x_i)) = O(kD)$,

Hence the overall worst case running time of the VH_{2DT} algorithm is $O(kD)$.

The Running time of $DIST_{2DT}$ Algorithm

The worst case running time of each step in the $DIST_{2DT}$ Algorithm is analyzed as follows:

- destination generation: $O(k)$,

- finding the shortest path: $O(Dk(\frac{N}{4} + \sum y_i + \sum x_i)) = O(DkN)$,

Hence the overall worst case running time of the $DIST_{2DT}$ algorithm is $O(DkN)$.

If the 2DT has the same number of nodes in every dimension, say $m = n$, then the total number of nodes is $N = m \times m = m^2$, and the maximum distance is $D = \frac{m+m}{2} = m$. Thus the complexity of the VH_{2DT} is $O(km)$, and the complexity of the $DIST_{2DT}$ is

$O(DkN) = O(mkm^2) = O(km^3)$. The complexities of 2DT are identical to those from the 2DM network.

4.3 Complexity of 3DM Network

The Running time of VH_{3DM} Algorithm

In the 3DM network ($m \times n \times p$), $N = m \times n \times p$ is the total number of nodes in the 3-dimensional mesh network, k is the number of destination nodes in the multicasting communication network, and the maximum distance is $D = m + n + p$. The worst case running time of each step is analyzed as follows:

- destination generation: $O(k)$,

- finding the path: $O(k(\sum_{i=1}^{p}(z_i) + \sum_{i=1}^{n}(y_i) + \sum_{i=1}^{m} x_i)) = O(kD)$,

Hence the overall worst-case running time of the VH_{3DM} algorithm is $O(kD)$.

The Running time of $DIST_{3DM}$ Algorithm

The worst case running time of each step in the $DIST_{3DM}$ Algorithm is analyzed as follows:

- destination generation: $O(k)$,

- finding the shortest path: $O(Dk(N + \sum z_i \sum +y_i + \sum x_i)) = O(DkN)$,

Hence the overall worst case running time of the $DIST_{3DM}$ algorithm is $O(DkN)$.

If the 3DM has the same number of nodes in every dimension, say $m = n = p$, then the total number of nodes is $N = m \times m \times m = m^3$, and the maximum distance is $D = m + m + m = 3m$. Thus the complexity of the VH_{2DM} is $O(kD) = O(k3m) = O(km)$, and the complexity of the $DIST_{2DM}$ is $O(DkN) = O(3mkm^3) = O(km^4)$. Evidently, the complexity of the $DIST_{3DM}$ algorithm is much greater than those from the VH_{3DM} algorithm.

4.4 Complexity of 3DT Network

The Running time of VH_{3DT} Algorithm

In the 3DT network $(m \times n \times p)$, $N = m \times n \times p$ is the total number of nodes in the 3-dimensional torus network, k is the number of destination nodes in the multicasting communication network, and the maximum distance is $D = \frac{m+n+p}{2}$. The worst case running time of each step is analyzed as follows:

- destination generation: $O(k)$,

- finding the path: $O(k(\sum_{i=1}^{p/2} z_i + \sum_{i=1}^{n/2} y_i + \sum_{i=1}^{m/2} x_i)) = O(kD)$,

Hence the overall worst case running time of the VH_{3DT} algorithm is $O(kD)$.

The Running time of $DIST_{3DT}$ Algorithm

The worst case running time of each step in the $DIST_{3DT}$ Algorithm is analyzed as follows:

- destination generation: $O(k)$,

- finding the shortest path: $O(Dk(\frac{N}{8} + \sum z_i + \sum y_i + \sum x_i)) = O(DkN)$,

Hence the overall worst case running time of the $DIST_{3DT}$ algorithm is $O(DkN)$.

Assume the 3DT has the same number of nodes in every dimension, say $m = n = p$, then the total number of nodes are $N = m \times m \times m = m^3$, and the maximum distance is $D = \frac{m+m+m}{2} = 1.5m$. Thus the complexity of the VH_{3DT} is $O(km)$, and the complexity of the $DIST_{3DT}$ is $O(DkN) = O(1.5mkm^3) = O(km^4)$.

It is clear that the VH_{2DM}, VH_{2DT}, VH_{3DM}, and VH_{3DT} all share the same time complexity of $O(kD)$. Meanwhile, the $DIST_{2DM}$, $DIST_{2DT}$, $DIST_{3DM}$, and $DIST_{3DT}$ have the same time complexity of $O(kDN)$. In addtion, the above concepts can be extended to the n-dimensional mesh or torus network. In general the VH algorithm has the time complexity of $O(kD)$ and the $DIST$ algorithm has the time complexity of $O(kDN)$. If we consider the n-dimensional mesh or the torus network $(m \times m \times \cdots \times m)$, the worst case running time of the VH algorithm is $O(knm)$, and the worst case running time of the $DIST$ algorithm is $O(knm^{n+1})$. Thus, the $DIST$ algorithm can not be used in high dimensional mesh and torus networks.

Chapter 5

Gossiping in the Multicasting Communication Environment

Gossiping is a fundamental communication problem. Initially, each node in a network holds some data, which must be routed so that in the end all nodes have the complete data (this problem is also called *all-to-all broadcast*). *Gossiping* is worth studying since it appears as a subroutine in many important problems.

5.1 Concept of Gossiping

Let N be any communication network (or graph) with $n \geq 4$ processors (nodes or vertices). The *broadcasting* problem defined over N consists of sending a message from one processor in the network to all the remaining processors. The *gossiping* problem over N consists of broadcasting n messages, each originating from a different processor. The gossiping communication model allows each processor to multicast one message to any subset of its adjacent processors, however no processor may receive more than one message at a time.

In the gossiping communication problem which contains n processors, initially, every processor holds one message and requires the remaining $n-1$ messages. Because a processor may receive only one message at a time, it is desirable to receive all messages in $n-1$ time units, which is the optimal lower bound of gossiping time. However, it is NP-complete to develop an algorithm for an arbitrary network, which can finish gossiping in $n-1$ time units. There are some special networks that can finish gossiping in $n-1$ time units. It is known that an n processor network with a Hamiltonian circuit can finish gossiping in $n-1$ time units (See Example 5.1). A circuit $x_0, x_1, \ldots, x_{n-1}, x_n, x_0$ (with $n > 1$) in a graph $G = (V, E)$ is called a Hamiltonian circuit if $V = \{x_0, x_1, \ldots, x_{n-1}, x_n\}$ and $x_i \neq x_j$ for $0 \leq i < j \leq n$. For the gossiping problem, it is not necessary for a network to have a Hamiltonian circuit to be solvable in $n-1$ steps. T.F. Gonzalez has proposed two networks that can achieve the gossiping problem in $n-1$ steps for $n = 6$ and $n = 10$ [7].

Example 5.1. *There are eight processors ($n = 8$) in network N (Figure 5.1), which includes a Hamiltonian circuit. The optimal schedule is for each processor to send its anticlockwise neighbor the message it holds, and then in the next six iterations, every processor keeps transmitting in an anticlockwise manner. It is simple to verify that all the communications can be carried out in $n-1$ steps, which is the best possible result.*

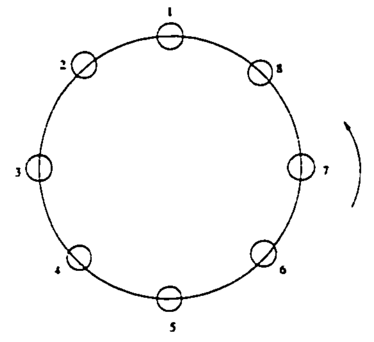


Figure 5.1: Network with a Hamiltonian circuit.

5.2 Communication Model

Now, I will formally define the communication model of the gossiping problem. Let N be any communication network (or graph) with $n \geq 4$ processors (nodes or vertices). Initially each processor P_i holds one message in its hold set and needs to receive the remaining $n-1$ messages. The multicasting communication model must satisfy the following restrictions:

1. During each time unit, each processor P_i may transmit one of the messages it holds, but such a message can be transmitted simultaneously to a subset of processors adjacent to P_i.

2. During each time unit each processor may receive at most one message.

The communication process ends when each processor has the complete n messages. Our problem consists of constructing a communication schedule with the least total communication time.

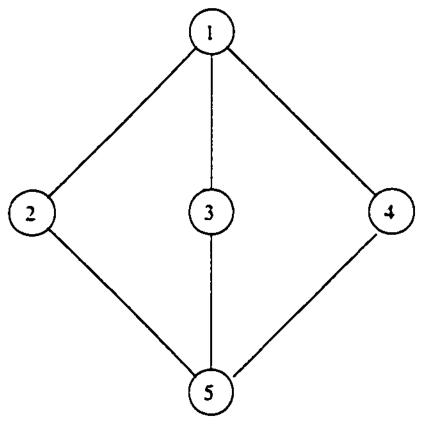


Figure 5.2: Network without a Hamiltonian circuit.

5.3 Gossiping Algorithm

The goal of the algorithm is to find a graph without a Hamiltonian circuit, which can also finish gossiping in $n-1$ time units. Let us first consider this example.

Example 5.2. *There are five processors* ($n = 5$) *in the network (Figure 5.2). Processor 1 and 5 are both connected to process 2,3,4; processor 2,3,4 are only connected to processor 1 and 5. This network does not contain a Hamiltonian circuit. A communication schedule with a total communication time equal to four is given in Table 5.1.*

time	Message: Processor $\to$ Processors		
T_1	$M_1 : P_1 \to P_{2,3,4}$	$M_2 : P_2 \to P_{1,5}$	
T_2	$M_2 : P_1 \to P_{3,4}$	$M_3 : P_3 \to P_{1,5}$	$M_5 : P_5 \to P_2$
T_3	$M_3 : P_1 \to P_{2,4}$	$M_4 : P_4 \to P_{1,5}$	$M_5 : P_5 \to P_3$
T_4	$M_4 : P_1 \to P_{2,3}$	$M_5 : P_2 \to P_1$ $M_1 : P_3 \to P_5$	$M_5 : P_5 \to P_4$

Table 5.1: Gossiping Schedule in a Network without Hamiltonian Circuit

Based on the example, a 5-processor-network can finish gossiping in 4 time units. Extend this example to a general case network, which has n processors in the network, and it can finish gossiping in $n-1$ time units. Now the algorithm is described in the following.

Let us define a network H. There are n processors in the network H. Processor 1 and n are connected to processors $2 \dots n-1$, and processor n is also connected to processor $2 \dots n-1$; processors $2 \dots n-1$ are only connected to processor 1 and n.

Proposition 5.1. *If a network contains a subgraph H, then this network can finish gossiping in $n-1$ time units.*

Proof. We will prove the proposition by describing an algorithm which finishes gossiping in $n-1$ time units.

Algorithm 5.1. *In a subgraph H, the communication schedule with a gossiping time equal to $n-1$ is given in Table 5.2.*

time	Message: Processor $\rightarrow$ Processors		
T_1	$M_1 : P_1 \rightarrow P_{2\ldots n-1}$	$M_2 : P_2 \rightarrow P_{1,n}$	
T_2	$M_2 : P_1 \rightarrow P_{2\ldots n-1}(except P_2)$	$M_3 : P_3 \rightarrow P_{1,n}$	$M_n : P_n \rightarrow P_2$
...	...	...	...
T_i	$M_i : P_1 \rightarrow P_{2\ldots n-1}(except P_i)$	$M_{i+1} : P_{i+1} \rightarrow P_{1,n}$	$M_n : P_n \rightarrow P_i$
...	...	...	...
T_{n-1}	$M_{n-1} : P_1 \rightarrow P_{2\ldots n-2}$	$M_n : P_2 \rightarrow P_1$ $M_1 : P_3 \rightarrow P_n$	$M_n : P_n \rightarrow P_{n-1}$

Table 5.2: Gossiping slgorithm schedule in subgraph H

In the gossiping problem, any processor P_j will receive all given messages m_i. Divide this problem into six different cases.

Case 1: If $i = 1$, and $2 \leq j \leq n-1$, then the processor p_j will receive m_i at time t_1 from the processor P_1.

Case 2: If $i = 1$, and $j = n$, then the processor p_n will receive m_1 at time $t_n - 1$ from the processor P_3.

Case 3: If $i = n$, and $2 \leq j \leq n-1$, then the message m_n will be sent to the processor P_2 at time t_2, the message m_n will be sent to the processor P_3 at time t_3, so the processor P_j will receive the message m_n at time t_j.

Case 4: If $i = n$, and $j = 1$, then the processor p_1 will receive the message m_n at time $t_n - 1$ from the processor P_2.

Case 5: If $2 \le i \le n-1$, and $j = 1, n$, then the message m_i will be sent to the P_1 and P_n at time t_{i-1}.

Case 6: If $2 \le i \le n-1$, and $2 \le j \le n-1$, in case 5 the processor P_1 will receive the message m_i at time t_{i-1}, then the P_1 will send the message m_i to the processor P_j at time t_i.

Overall, the above analysis can guarantee that any processor P_j will receive any given message m_i, and receive them within $n-1$ time units. □

If there exists such a subgraph in an arbitrary graph, evidently, it can finish gossiping in $n-1$ time units. How would we find such a subgraph in a given graph? Let us use the adjacency matrix to solve this problem. The adjacency matrix is the $n \times n$ zero-one matrix with 1 as its (i,j)th entry when v_i and v_j are adjacent, and 0 as its (i,j)th entry when they are not adjacent. In other words, if its adjacency matrix is $A = [a_{i,j}]$, then

$$a_{i,j} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0, & \text{otherwise.} \end{cases}$$

Algorithm 5.2. *Consider an arbitrary graph with n nodes. Using the adjacency matrix to represent this graph.*

1. *Find a row i, which contains at least $n-2$ 1s.*

2. *Find another row j, which contains $n-2$ 1s except in column i.*

If these conditions are satisfied, there must be such a subgraph.

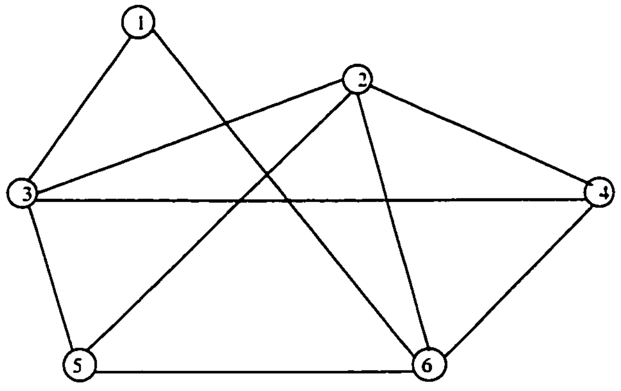


Figure 5.3: Network containing subgraph H

Example 5.3. *Let us consider an arbitrary network with 6 nodes as shown in Figure 5.3. First contruct the matrix.*

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

The second row has four 1s, but there is no other row which has the same pattern as row 2. As we continue searching, row 3 and 6 have four 1s, and the 1s are in columns 1, 2, 4, and 5. Thus, there is a desirable subgraph. As a result, this network can finish gossiping in five time units.

Chapter 6

Conclusion and Future Work

This thesis presents multicasting algorithms for 2- and 3-dimensional mesh and torus networks. In general, the VH algorithm follows the dimension ordered routing. Routing in a particular dimension is always complete before proceeding to the next dimension. The $DIST$ algorithm follows the shortest path to the existing multicasting tree to reduce the total traffic. The VH algorithm has a worst case time complexity of $O(kD)$ for any dimensional mesh and torus networks. The $DIST$ algorithm has a worst case time complexity of $O(kDN)$ for any dimensional mesh and torus networks, where k is the number of destination nodes, D is the maximum distance for any destination node, and N is the total number of nodes in the network.

Time and traffic are two important parameters used when evaluating the performance of multicasting algorithms. The VH algorithm delivers the message in the minimal time, while the $DIST$ algorithm generates less traffic than the VH algorithm.

Not only can the VH and the $DIST$ algorithms be used in the mesh and the torus networks, but also can they be applied to the hypercube network.

Gossiping communication is an important communication model. It is desirable to finish gossiping in $n-1$ time units for a n node network. However, it is NP-complete to develop such an algorithm for an arbitrary network. A special type of a network is shown, which can finish gossiping in $n-1$ steps.

The emphasis of this thesis is on the theoretical aspect of the multicasting communication. There are many more issues which certainly require further investigation. First, the upper bound of $DIST$ algorithm should be more tight. The upper bound of the $DIST_{2DM}$ is $3D_{max}$ in this thesis, which can be reduced in future study. Second, other better heuristic routing algorithms should be studied, which minimize the time while keeping the traffic at a minimum. Third, the deadlock issue should be concerned that may be caused by the multicasting communication. Fourth, the faulty node in the communication network should be considered when developing a routing algorithm because the faulty-tolerant routing scheme is a key to the performance of reliable network communication. Finally, we should study other evaluation criteria based on different underlying switching techniques, such as the multicasting algorithm in wormhole routing.

Bibliography

[1] T.S. Chen, N.C. Wang, and C.P. Chu, *Multicast Communication in Wormhole-Routed Star Graph Interconnection Networks*, Parallel Comupting **26** (2000), 1459–1490.

[2] R. Cypher, and L. Gravano, *Storage-Efficient, Deadlock-Free Packet Routing Algorithms for Torus Networks*, IEEE Transactions on Computers **Vol.43, No.12** (1994), 1376–1385.

[3] P. Fraigniaud and E. Lazard, *Methods and problems of communication in usual networks*, Discrete Applied Mathematics **53** (1994), 79–133.

[4] S. Fujita and M. Yamashita, *Optimal Group Gossiping in hypercubes Under a Circuit-Switching Model*, SIAM J. on Computing **Vol.25, No.5** (1996), 1045–1060.

[5] T.F. Gonzalez, *Simple Algorithms for Multimessage Multicasting with Forwarding*, Algorithmica **29** (2001), 511–533.

[6] T.F. Gonzalez, *Gossiping in the Multicasting Communication Environment*, "Proceedings of the International Parallel and Distributed Processing Symposium," IPDPS 2001, (6 pages).

[7] T.F. Gonzalez, *An Efficient Algorithms for Gossiping in the Multicasting Communication Environment*, (2001), 14 pages.

[8] T.F. Gonzalez, *Complexity and Approximations for Multimessage Multicasting*, J. Parallel and Distributed Computing **55** (1998), 215–235.

[9] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1972.

[10] H. Harutyunyan and X.L. Liu, *Multicast Algorithm in Torus Network*, Twenty-First IASTED International Multi-Conference, Applied Informatics (AI 2003), 378–381.

[11] J. Hayes, T. Mudge, Q. Stout, S. Colley and J. Palmer, *Architecture of a Hypercube Supercomputer*, Proc. 1986 International Conference on Parallel Processing (1986), 653–660.

[12] Y. Lan, *Adaptive Fault-Tolerant Multicast in Hypercube Multicomputers*, J. Parallel and Distributed Computing **23** (1994), 80–93.

[13] Y. Lan, A.H. Esfahanian and L.M. Ni, *Multicast in Hypercube Multiprocessors*, J. Parallel and Distributed Computing **8** (1990), 30–41.

[14] X. Lin and L.M. Ni, *Multicast Communication in Multicomputer Networks*, IEEE Transactions on Parallel and Distributed Systems **Vol.4, No.10** (1993), 1105–1117.

[15] P. Mieghem, G. Hooghiemstra, and R. Hofstad, *On the Efficiency of Multicast*, IEEE/ACM Transactions on Networking **Vol.9, No.6** (2001), 719–732.

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

[17] 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** (1995), 1029–1042.

[18] Y. Saad and M.H. Schultz, *Topological properties of hypercubes*, IEEE Transactions on Computers **Vol.37, No.7** (1988), 867–872.

[19] H. Shen, *Efficient Multiple Multicasting in hypercubes*, J. Systems ARxhitecture **43** (1997), 655–662.

[20] H. Shen, D.J. Evans and J. You, *Fault-Tolerant Multicast with Traffic-Balancing in Hypercubes*, Parallel Algorithms and Applications **Vol.11** (1997), 287–298.

[21] J.P. Sheu and M.Y. Su, *A Multicast Algorithm for Hypercube Multiprocessors*, Parallel Algorithms and Applications **Vol.2** (1994), 277–290.

[22] J. Wu, *Maximum-Shortest-Path (MSP): An Optimal Routing Policy for Mesh-Connected Multicomputers*, IEEE Transactions on Reliability **Vol.48, No.3** (1999), 247–255.