A HEURISTIC FOR BROADCASTING IN ARBITRARY NETWORKS

Bin Shao

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

December 2002

Abstract

A Heuristic for Broadcasting in Arbitrary Networks

Bin Shao

Finding the optimal broadcasting schedule in an arbitrary network is an NP-hard problem. Many papers have been published on the topic of finding an efficient and effective heuristic for broadcasting. In this thesis, a heuristic algorithm for broadcasting in arbitrary networks is presented. This heuristic is first tested in several commonly used graphs; such as $Butterfly$, CCC_d, $deBruijn$ and $Shuffle\ Exchange$. In these graphs, this heuristic generates almost the same broadcast schedules as the best existing heuristic for broadcasting. However, the time complexity of this heuristic is much lower. The heuristic is also tested in three of the best-known network design models, and the heuristic outperforms the best existing heuristic in these models. The time complexity of this heuristic is $O(R \cdot m)$, where R represents the rounds of broadcasting, and m stands for the total number of communication links in the network.

Acknowledgments

I sincerely appreciate the help of my supervisor, Hovhannes A.Harutyunyan, one who has guided me along since the very beginning. His charisma has inspired me throughout the course of my learning, and will continue to do so.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Broadcasting is a one to n information dissemination problem in networks. The problem of determining the optimal broadcast time for an aribitrary node in an arbitrary graph G is NP-complete. A new heuristic for broadcasting is briefly introduced in this chapter.

1.1 Problem Statement

Computer networks, from local area networks to the Internet, have become essential in several aspects of modern society. The performance of information dissemination in networks often determines their overall efficiency. Broadcasting is one such information dissemination problem.

Communication efficiency becomes particularly important when the network supports a distributed file or database system. There are two approaches to reduce the delay of information dissemination: one is to reduce the amount of data being transferred, while the other is to minimize the delay of information spreading [24]. The new heuristic introduced in this thesis can be used in the latter approach.

In addition, broadcasting is a vital communication problem in multiprocessor computer systems. There are many problems that could not be solved by a single processor in an acceptable amount of time. One solution is to divide the problem into subproblems that can be performed in parallel. A single processor handles one of these subproblems. The results of certain subproblems must be transferred among these processors for further computing [22]. The performance of the communication often determines the efficiency of the whole parallel system. Therefore, broadcasting quickly is an important goal in parallel systems.

A network can be modeled as a graph $G = (V, E)$, where V is the set of vertices (or nodes) and E is the set of edges (or communication lines). Two nodes $u \in V$ and $v \in V$ are *adjacent* if there is an edge $e \in E$, such that $e = (u, v)$. We also say node u or v is a *neighbour* of another node. The *degree* of a node is the number of neighbours of this node. The *degree* of a graph G is the maximum degree among all nodes in this graph. Δ stands for the degree of a graph. A path p in a graph $G = (V, E)$ is a sequence of nodes of the form $p = v_1, v_2, ..., v_n, (n \geq 2)$, in which each node v_i is adjacent to the next node. Obviously, the path p is also a sequence of edges. The length of a path is the number of edges in the path. The length of the

shortest path between two nodes is the *distance* between them. The *diameter* of a graph is the maximum of the distances between all pairs of nodes in the graph. A graph $G = (V, E)$ is said to be *connected* if there is a path between any two nodes on G. It is natural to assume that the network is represented by a connected graph.

The study of the information dissemination problem occurs when the following problem is raised: "There are n ladies, and each one of them knows an item of scandal that is not known to any of the others. They communicate by telephone, and whenever two ladies make a call, they pass on to each other, as much scandal as they know at the time. How many calls are needed before all ladies know all the scandal?" [13] This problem, which has become known as the *Gossip Problem*, or the *Telephone Problem*, has in turn been the source of dozens of research papers that have concerned the spread of information among a set of people, whether it be by telephone calls, conference calls, letters or even computer networks.

Broadcasting in a network is the process in which a node in the network sends a message to all other nodes. The main difference between broadcasting and gossiping is that, in gossiping, each node has different messages that will be sent to all others during the process, while only one node has one message to send to all other nodes in broadcasting.

Communications in networks could be classified into three types, depending on where the communication bottleneck occurs [10].

(1) If, during communication, a processor can only use one of its links, we call this

situation *processor-bound* because processors cannot quickly relay messages and will hamper the efficiency of the network. This pattern is also called *1-port* or *whispering.*

(2) On the contrary, when a processor can use all of its links at the same time, communications are said to be *link-bound*, because it is now the number of links that limits communications. This pattern is also called *n-ports* or *shouting.*

(3) Between these two extreme possibilities, we have the case where a processor can only use k links at the same time.

In this thesis, we consider the broadcasting problem in the *constant* model with a processor-bound constraint. In the *constant* model, the time of communication between two nodes is equal to one time unit. The constant model is popular, because it is simple, and it models communication efficiently when small messages are exchanged.

At the beginning of broadcasting, only one node u of V, called the *originator*, is informed. During each unit of time, each informed node passes the message to only one of its uninformed adjacent nodes. Such an action is denoted as a *call.* Broadcasting can be modeled as a sequence of parallel calls between nodes in graph G.

The model considered in this thesis has the following constraints:

(1) A node can only call one adjacent node per unit of time.

(2) A node can participate in only one call per unit of time.

(3) Each call requires one unit of time.

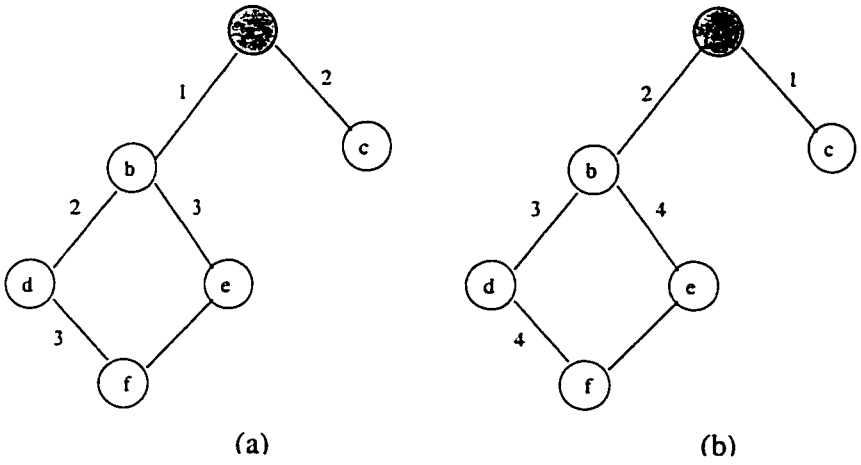


Figure 1: Examples of Broadcast Schedule

Many calls can be performed in parallel. A *round* is a set of parallel calls in the same time unit. We use the number of rounds to measure the broadcast time. Since one round spends one unit of time, the number of rounds is equal to the total time-steps needed for broadcasting. Given a graph G, the broadcast time $b(G, u)$, or simply $b(u)$, is the minimum broadcast time of graph G originated by node u. The broadcast time of graph G is defined as follows: $b(G) = max\{\, b(G, u) \mid u \in V\}$.

The two example graphs in Figure 1 present the above concept. In each graph, the node with the shadowed background is the originator. The graph in Figure 1 (a) represents the optimal broadcast schedule in this graph originated by node a, while the broadcast schedule shown in Figure 1 (b) is not optimal. The number beside an edge indicates the *round* in which the edge is used. For instance, in Figure 1 (a), the calls a to c and b to d compose round 2. Also, calls only occur on the edges marked by the numbers. By the definition, $b(a) = 3$.

1.2 NP-completeness

The problem of finding the optimal broadcasting time for an arbitrary node u in an arbitrary graph G is NP-complete. This statement has been proved in [25]. The crux of the proof is to prove that the broadcast problem is equivalent to another NP-complete problem: the three-dimension matching (3DM) problem.

3DM: Given a set $M \subseteq X \times Y \times Z$, where X, Y and Z are disjoint sets having the same number of m elements. Does there exist a subset $N \subseteq M$ such that $\mid N \mid = m$ and no two elements of N agree in any coordinate? This problem has been proven to be NP-complete in [12].

In [25], a more general problem named BROADCAST TIME is presented first:

BROADCAST TIME. Given a graph $G = (V, E)$, a set of nodes $V_0 \subseteq V$ and a positive integer k, is there such a sequence $V_0, E_1, V_1, E_2, V_2, \cdots, E_k, V_k$, that applies to the following conditions?

1. $V_i \subseteq V$, for $0 \leq i \leq k$.

2. $E_i \subseteq E$, for $1 \leq i \leq k$.

3. For every edge in E_i, only one of end node of edges in E_i belongs to V_{i-1}.

4. $V_i = V_{i-1} \cup \{v : uv \in E\}$

5. $V_k = V$

V_i is the set of nodes who are informed at time by accepting the message through

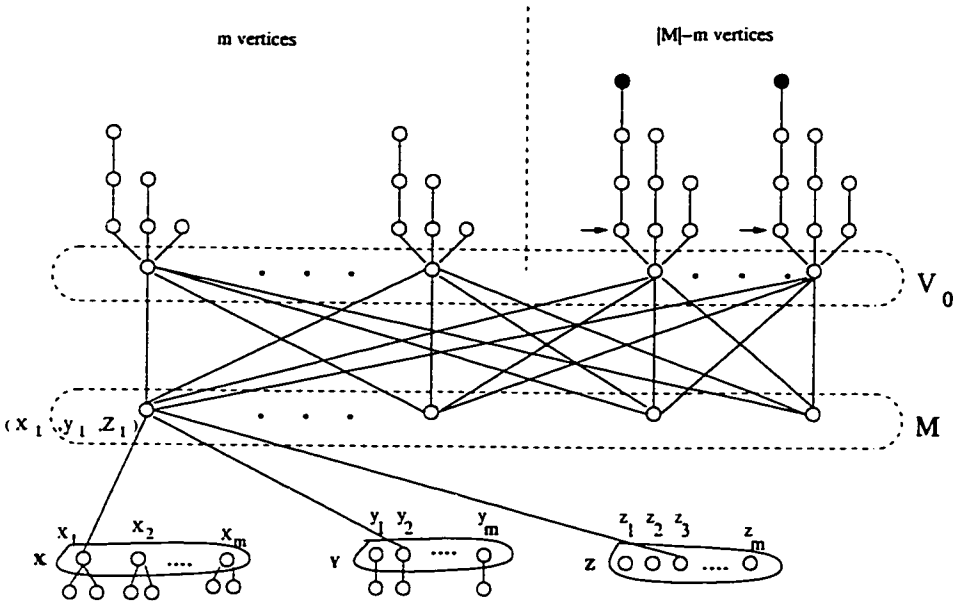


Figure 2: The graph G corresponding to the problem 3DM

edges in E_i. Such a problem is just like a broadcast problem with more than one originator. It is obvious that when $| V_0 | = 1$, this general problem reduces to determining $b(G, u)$.

A solution to the problem 3DM is equivalent to a solution of the BROADCAST TIME problem with $k = 4$ in a graph G constructed in [25]. The graph G is shown in Figure 2. The independent sets V_0 and M are equal in size. The nodes in V_0 and M compose a complete bipartite graph, which is a subgraph of G. If $(x_i, y_j, z_k) \in M$, then the corresponding node of M in G is joined to the nodes x_i of X, y_j of Y and z_k of Z. For the purposes of illustration, (x_i, y_j, z_k) is assumed to be an element of M. Other nodes and edges of G are precisely as depicted.

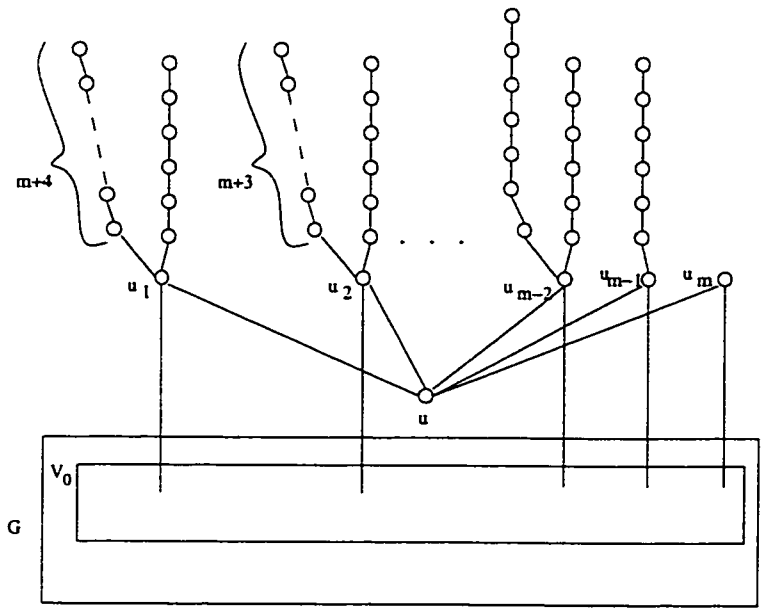


Figure 3: The extended graph H

Suppose that there is a solution to BROADCAST TIME in G. In order to inform the nodes of G indicated by solid circles in time 4, $|M|-m$ nodes of V_0 must broadcast in an upward direction to the nodes pointed by the arrow. In order for X, Y, Z and the bottom line to be informed in time 4, the m nodes of V_0 must broadcast to a subset S of M at time 1. The size of S is m. Then, the nodes in S must broadcast to distinct elements of X, Y and Z at times 2, 3 and 4 respectively. This is possible if and only if S is a solution to 3DM.

The graph G is further extended to the graph H (shown in Figure 3) as follows. First, an independent set $U = \{u_1, u_2, \cdots, u_m\}$ and a node u are added to G. Then, append to u_i $m-1$ paths. The lengths of those path are $6, 7, \cdots, m+5-i$ respectively. Finally, m edges are added to create a matching between U and V_0.

We can see that a solution to the broadcast time problem with $k = 4$, and V_0 the independent node set of size $m \geq 2$ in G, is equivalent to determining if $b(H, U) = m+5$ in H. So, determining if $b(H, U) = m+5$ in H is equivalent to 3DM. Therefore, determining the broadcasting time of a arbitrary node in an arbitrary graph is NP-complete.

1.3 A New Heuristic

It is of a major interest to derive a polynomial heuristic for broadcast because finding optimal $b(G, u)$ for a arbitrary graph is NP-complete. Since a heuristic is presented in [24], many papers on this topic have been published. Algorithms solving both broadcasting and gossiping are introduced in [11]. The crux of the algorithm for gossiping in [11] is finding a maximum weight matching in each round because the process of gossiping can be looked at as a sequence of matchings between both informed and uninformed nodes. In [3], an improved algorithm was introduced by using a different way to weight the nodes. The algorithm in [3] is the best existing heuristic for broadcasting in practice.

In this thesis, a new heuristic is presented. The basic idea with regards to the new heuristic is to find a match between the set of informed and uninformed nodes in each round, and then to distribute messages between them. To achieve this, in each round, the heuristic first perfoms a modified BFS (breadth first search) from all informed nodes with uninformed neighbours towards all uninformed nodes. During

the process, all uninformed nodes are labeled with the shortest distance from it to any informed node. If a node u is labeled by i, then all nodes labeled by $i+1$ are the *children* of u. The *descendant graph* of node u, $DG(u)$ is composed by u and all its descendants. In this heuristic, a way is designed to estimate the broadcast time of $DG(u)$, and this time is assigned as the weight of u. After that, the heuristic generates a match between the set of informed nodes and uninformed nodes. In this match, the number of pairs of nodes is at a maximum, and given this, the sum of weights of matched nodes is also at a maximum. Finally, every matched informed node sends the message to its mate.

The new heuristic was first tested on several commonly used graphs, such as the *deBruijn*, the *Shuffle Exchange*, the *Butterfly* and the CCC_d graph. For these graphs, this heuristic works as well as that from [3]. Then the heuristic was tested on three graph models from a network simulator: ns-2. The first model is one of the best-known network design models to test routing algorithms in network such as **IP** and **ATM** network. The second one is a well-known network design model for the Internet. The third model generates pure random networks. The new heuristic outperforms best known broadcast algorithms in these models. For only one graph out of two hundred tested graphs, the performance of the new heuristic is worse than that of the heuristic in [3]. In forty percent of the cases, the new heuristic gives a better broadcast time than other algorithms. Another improvment of the new heuristic is the time complexity. The time complexity of one round of the new heuristic is $O(m)$, while the complexity of one round of the algorithm from [3] is $O(n^2 \cdot m)$, where n is

the number of nodes and m is the number of edges in a graph.

1.4 Thesis Outline

The remainder of this thesis is structured as follows: the commonly used topologies, previously-known heuristics and three graphs models are introduced in Chapter 2. This background is used in the later chapters. A new heuristic is formally presented in Chapter 3. The theoretical and experimental results concerning the heuristic are introduced in Chapter 4 and 5, respectively. The performance of the new heuristic is compared with those from several previously-known heuristics in Chapter 6.

Chapter 2

The Background: Usual Topologies, Previously-known Heuristics and Graph Models

In the first section, some usual topologies and the results on broadcasting in these topologies are introduced. Several previously-known heuristics for broadcasting are briefly introduced in Section 2. In the last section, three commonly used graph models are presented. These models are used to test the heuristic.

2.1 Usual Topologies

2.1.1 Arbitrary Graph

It is well known that, for any graph on n nodes, $\lceil log_2 n \rceil \leq b(G) \leq n-1$ [10]. Because any node holding a message could only send it to one of its adjacent nodes, the number of informed nodes could at most be doubled in each round. Thus, at least $\lceil log_2 n \rceil$ rounds are needed for broadcasting. On the other hand, at least one node must be informed in each round. A situation in which no new node is informed means that the broadcasting has been finished. Therefore, broadcasting takes at most $n-1$ rounds.

For any graph of maximum degree Δ, and diameter D, where $D \leq b(G) \leq \Delta D$, because it is possible to broadcast in any shortest path spanning tree of G of height D and maximum degree Δ in at most ΔD rounds. In [10], the following lemma is proven.

Lemma 2.1 *In any graph of diameter D, if three different nodes u, v_1 and v_2, with both v_1 and v_2 at a distance D from u, exist, then $b(G) \geq D+1$.*

This will be used when discussing broadcasting time in several topologies.

2.1.2 Usual Topologies

Many commonly used topologies and their broadcast times are presented in the three surveys: [10], [15] and [17]. Some important topologies are elaborated in [20]. The theoretical and testing results converning the new heuristic in these topologies will

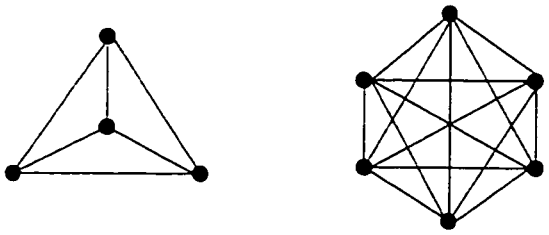


Figure 4: Complete graphs for N=4 and N=6

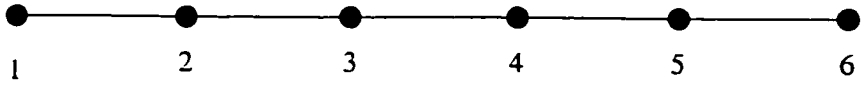


Figure 5: The Path Graph for N=6

be presented later.

- *The complete graph* K_N: All N nodes are linked together (see Figure 4) [10]. Each node has degree $N-1$. The diameter is one and the number of edges is $N(N-1)/2$. It is easy to see that $b(K_N) = \lceil log_2 N \rceil$. Given that any informed node sends the message to any of its uninformed neighbours in each round, the former bound could be obtained.

- *The Path* P_N: The path of length n, denoted by P_N, is the graph whose nodes are all labeled by integers from 1 to n, and whose edges connect the node labled by integer i $(1 \leq i \leq n)$ with the node labeled by $i+1$. P_N has n nodes, a diameter of $n-1$ and a maximum degree of 2 (see Figure 5).

Suppose a node u in P_N is the originator. Assume the distance of the node u to the two ends of P_N are l_1 and l_2. The reasonable broadcast scheme is such that: in the first

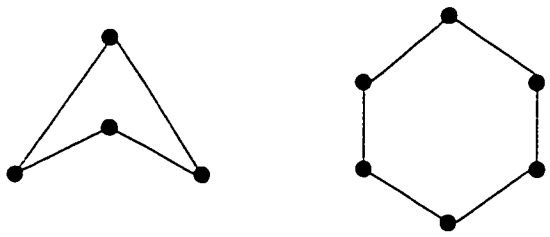


Figure 6: Ring graphs for N=4 and N=6

step, the originator sends the message to the end with a longer distance; then, in the later steps, the two nodes that have uninformed neighbours send the message to their respective neighbours. Consider any node u in P_N, $b(u, P_N) = max\{l_1, l_2+1\}$, $l_1 \geq l_2$. When, u is at either end of P_N, $b(u, P_N)$ is at the maximum. In such a case $b(u, P_N)$ $= b(P_N) = n - 1$.

- *Ring graph* C_N: Each node is linked to only two neighbours, thus the degree is two (see Figure 6) [10]. There is only one reasonable broadcast scheme: the originator sends the message to one of its neighbours, then at each round, the two nodes that have uninformed neighbours send the messages to them. Therefore:

$$b(C_N) = \lceil \frac{N}{2} \rceil = \begin{cases} D & if\ N\ is\ even \\ D+1 & otherwise \end{cases} \quad (1)$$

- *The d-grid graph* $GD_N = P_{n_1} \Box \cdots \Box P_{n_d}$: For $1 \leq i \leq d$, P_{n_i} is a path on n_i nodes. The *d-grid* graph is also called *mesh*. A *2-grid* is shown in Figure 7. The following bound is shown in [9].

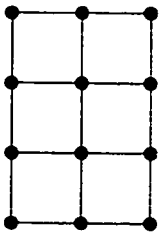


Figure 7: A 2-grid graph with 12 nodes

$$b(P_{n_1} \square \cdots \square P_{n_d}) = \sum_{i=1}^{d} n_i - d = D \tag{2}$$

In [9], the authors also pointed out the follows:

Let v be a node in a 2-grid graph $G_{m,n}$, where m and n are the number of nodes in the paths.

a) When v is a corner node, $b(v) = m + n - 2$.

b) If v is a side node, then $b(v)$ is equal to the maximum distance from v to a corner node.

c) If v is an interior node at position (i, j), then $b(v)$ is equal to the maximum distance from v to a corner node

plus 1 if $j = \frac{n+1}{2}$, or

plus 2 if $j = \frac{n+1}{2} = i = \frac{m+1}{2}$.

• *Hypercube graph* H_d: The $r - dimensional$ hypercube has $N = 2^r$ nodes and $r2^{r-1}$ edges. Each node corresponds to an r-bit binary string, and two nodes are

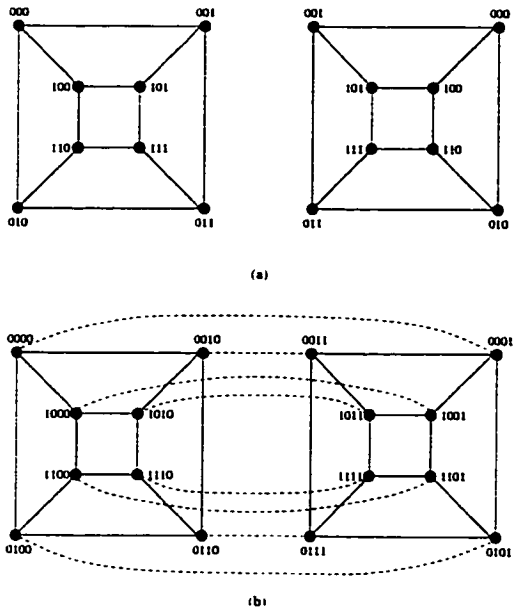


Figure 8: Hypercube graphs

linked with an edge if and only if their binary strings differ by precisely one bit. As a consequence, each node is adjacent to $r = logN$ other nodes, one for each bit position. Any $r - dimensional$ hypercube could be derived from two $(r-1) - dimensional$ hypercube graphs [20]. Figure 8 presents the construction of a $4 - dimensional$ hypercube (b) from two $3 - dimensional$ hypercubes (a). Dashed edges form a match between the two $3 - dimensional$ cubes. The diameter of H_d is d. To broadcast in time $\lceil log_2 N \rceil$, use the following scheme: at step i, each informed node sends the message in dimension i $(1 \leq i \leq d)$.

- *Cube connected cycles* CCC_d : The CCC_d (see Figure 9) is a modification of H_d. The $r - dimensional$ CCC_r is constructed from the $r - dimensional$ hypercube by replacing each node of the hypercube with a cycle of r nodes in the CCC_r. The

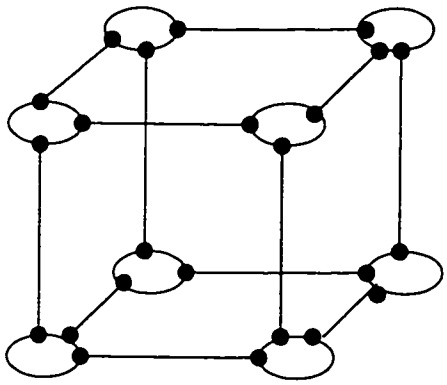


Figure 9: The CCC_3 graphs

diameter of the CCC_d is $2d+\lfloor d/2 \rfloor -2$, where $d > 3$ [21]. A straightforward algorithm gives a broadcast time of $\lceil \frac{5d}{2} \rceil - 1$ [10]: first relay the message to the hypercube neighbour. then to the right neighbour on the ring, then to the left one.

(1) If d is even, then $\lceil \frac{5d}{2} \rceil - 1 = D + 1$. Therefore

$$D \leq b(CCC_d) \leq D + 1 \tag{3}$$

with the exception of $d = 4$, where Lemma 2.1 applies, and therefore

$$b(CCC_4) = D + 1 \tag{4}$$

(2) If d is odd: $\lceil \frac{5d}{2} \rceil - 1 = D + 2$. Lemma 2.1 still applies, and therefore

$$D + 1 \leq b(CCC_d) \leq D + 2 \tag{5}$$

• *The butterfly graph* BF_d: The $r - dimensional$ butterfly BF_d has $r2^r$ nodes (see Figure 10). Each node is labeled with a pair of numbers (l, x). l represents the level ($0 \leq l \leq d - 1$), and $x = x_0 \cdots x_{d-1}$ is a d-bits binary string called the

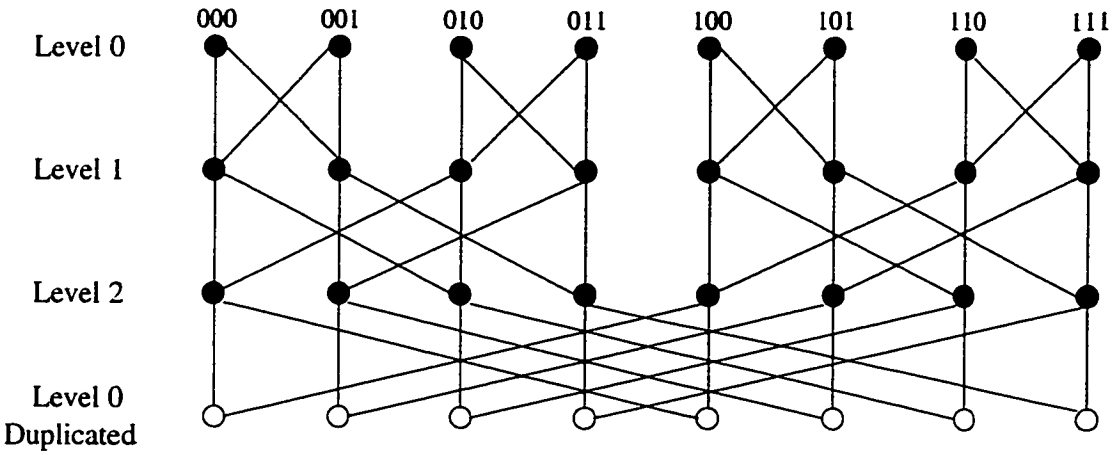


Figure 10: The BF_3 graphs

position − within − level. Two nodes (l_0, x_0) and (l_1, x_1) in BF_d are linked by an edge if and only if $l_1 = l_0 + 1 \ mod \ d$ and either if $x_0 = x_1$ or if x_0 and x_1 differ by the l_0th bit. BF_d has degree four and diameter $\lfloor 3d/2 \rfloor$.

The authors showed in [18] that:

$$1.7417d \leq b(BF_d) \leq 2d - 1 \tag{6}$$

The broadcast time for a butterfly graph with a dimension in the range from 2 to 20 is also given in [18]. These results are shown in Table 1. The results from Table 1 will be used in Chapter 5.

• *The de Bruijn graph UB(d, D)*: The de Bruijn digraph $B(d, D)$ with indegree and outdegree d and diameter D, is the digraph whose $N = d^D$ nodes is denoted by the words of length D on an alphabet of d letters. There is a direct edge from each node $(x_0 x_1 \cdots x_{D-1})$ to $(x_1 \cdots x_{D-1} 0)$ and $(x_1 \cdots x_{D-1} 1)$. The de Bruijn graph

d	Lower bound	Upper bound	N
2	3	3	8
3	5	5	24
4	7	7	64
5	8	9	160
6	10	11	384
7	11	13	896
8	13	15	2048
9	15	17	4608
10	16	19	10240
11	18	21	22528
12	19	23	49152
13	21	25	106496
14	23	27	229376
15	24	29	491520
16	26	31	1048576

Table 1: Broadcast time for small butterfly networks

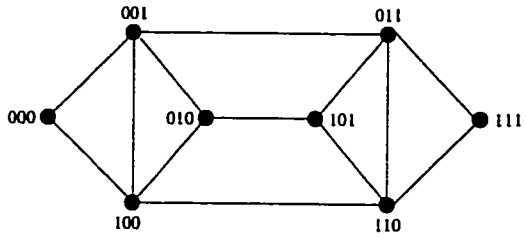


Figure 11: The $UB(2,3)$ graph

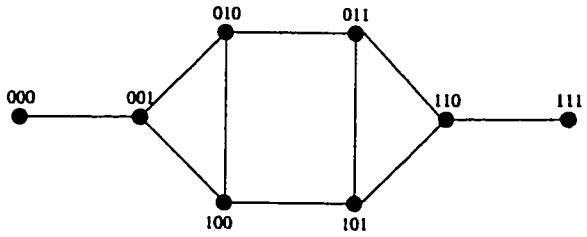


Figure 12: The SE_3 graphs

$UB(d, D)$ is obtained by removing the edge orientation in $B(d, D)$. In this thesis, $UB(2, D)$ is considered (see Figure 11).

In [18], the following result is proven: $1.3171D \leq b(UB(2, d))$.

The upper bound is shown in [5]: $b(UB(2, D)) \leq \frac{3}{2}(D + 1)$.

• *The shuffle-exchange graph* SE_d: The $d - dimensional$ shuffle-exchange graph has $N = 2^d$ nodes (see Figure 12). Each node corresponds to a unique d-bit binary number, and two nodes u and v are linked by an edge, if either

1) u and v differ in precisely the last bit, or

2) u is a left or right cyclic shift of v.

The broadcasting time for SE_d is $2d - 1$ [7], which is equal to the diameter of SE_d.

• *The star graph* S_n: The n-Star Graph is the Cayley graph on the group G consisting of all permutations on n symbols, and the set of generators g defined as follows: The set g consists of $n - 1$ transpositions $\{g_2, g_3, g_4, \cdots, g_n\}$ where g_1 is the transposition that switches the ith element with the first, and leaves the remaining elements in their original positions (see Figure 13). In [1], it is proven that the diameter of the star graph is $\lfloor 3(n - 1)/2 \rfloor$. The following result is shown in [6]:

$$\lceil log_2 N \rceil \leq b(S_n) \leq \lceil log_2 N \rceil + \lceil 7n/4 \rceil + \lceil log_2 n \rceil \qquad (7)$$

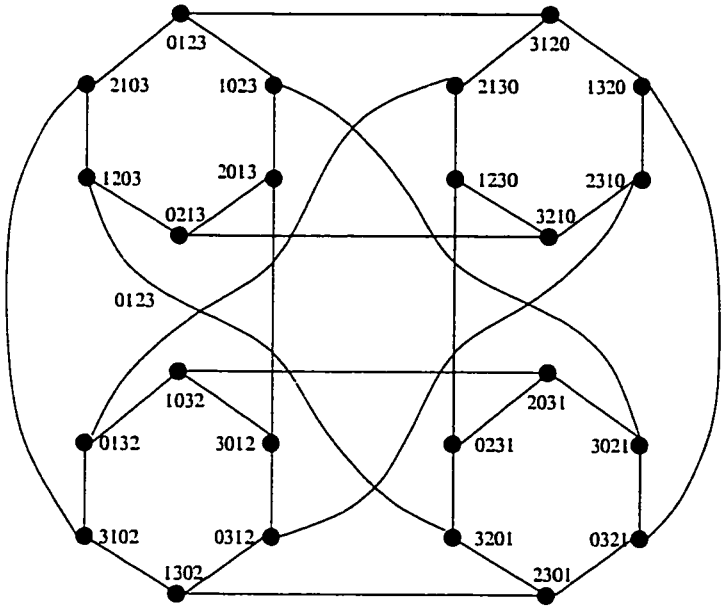


Figure 13: The star graph S_4

2.2 Previously-known Heuristics

In this section, several previously-known heuristics will be introduced. The heuristic in [3] is the best existing heuristic. The heuristics in [19] and [23] give the theoretical results for broadcasting time. In Chapter 6, the results of the new heuristic will be compared with the results of these three previously-known heuristics.

2.2.1 Round_Heuristic

The heuristic in [3], named Round_Heuristic, is the best heuristic in practice. This heuristic is designed for both gossiping and broadcast. Only its performance in broadcasting is considered in this thesis. In [3], the testing results from this heuristic in

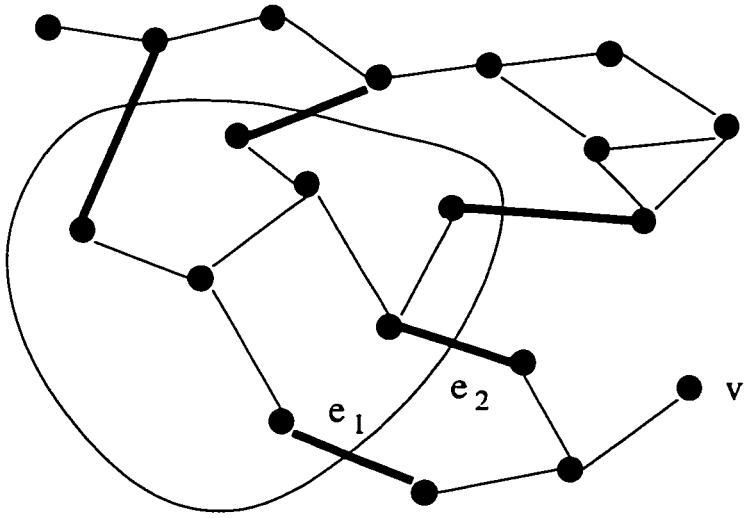


Figure 14: The Definitions in Round-Heuristic

several commonly used graphs were presented. Most of the results are equal to the optimal broadcast time.

In each round of broadcasting, weights are assigned to all edges in the graph. Then, a *maximum weighted matching* is constructed in the graph. The matched edges are active in this round. When an edge is active in a round, it means that a message is passed through this edge in this round.

The most important part of Round_Heuristic is setting the weight. There were two approach used in the Round_Heuristic. One is the *Potential Approach*, wherein the weight of an edge (v, w) is set to equal its *potential*, defined as the number of messages known by either v or w, but not by both of them. In broadcasting, the weight could only be 0 or 1. Although this approach is simple, requires little storage and is very

fast, it performs worse than the second approach: the Breadth-First-Search (BFS) approach.

Several definitions are needed to introduce the BFS approach. The dispersion region $DR(p,t)$ of a message p refers to the set of nodes that know p at the beginning of round t (this is a connected subgraph). For a node v, $dist_v(p,t)$ denotes the shortest distance in the graph from v to a node $w \in DR(p,t)$. The set of border-crossing edges $bce(p,t)$ is defined as $bce(p,t) = \{(v,w) \in E \mid v \in DR(p,t) \text{ and } w \notin DR(p,t)\}$. For a node $v \notin DR(p,t)$, $bce_v(p,t)$ consists of all edges in $bce(p,t)$ that lie on the shortest path from $DR(p,t)$ to v. Figure 14 illustrates these definitions. In this figure, the edges of $bce(p,t)$ are drawn in bold. $dist_v(p,t) = 3$ and $bce_v(p,t) = \{e_1, e_2\}$.

When considering an edge $e \in bce(p,t)$, how useful is e for the rapid dissemination of p? Message p should be routed on shortest paths from $DR(p,t)$ to all other nodes. If e lies on more of these shortest paths, it is more useful. The larger $dist_v(p,t)$ is, the more priority should be given to forwarding p towards v. These criterion motivate the using of two parameters $Dist\_Exp$ and Num_Exp in calculating the weight. In round t, every node $v \notin DR(p,t)$ attributes the weight to every edge $e \in bce_v(p,t)$ as follows:

$$weight(v,p,t) = \frac{dist_v(p,t)^{Dist\_Exp}}{\mid bce_v(p,t) \mid^{Num\_Exp}} \tag{8}$$

In [3], a modified breadth first search algorithm is used to calculate the weight. Because it is a breadth first search algorithm, the nodes are considered in an order of increasing $dist_v(p,t)$. For nodes v with $dist_v(p,t) = 1$, $bce_v(p,t)$ consists of all

adjacent edges that connect v to a node in $DR(p,t)$. For larger $dist_v(p,t)$, the algorithm computes the *union* of the sets $bce_{w_i}(p,t)$, for all nodes w_j adjacent to v with $dist_{w_i}(p,t) = dist_{w_i}(p,t) - 1$. Thus, the calculation of $bce_v(p,t)$ can easily be incorporated into the BFS search.

For a node v, $bce_v(p,t)$ is the union of a maximum of n sets with a maximum of m elements each. This computation takes $O(n \cdot m)$ time. The $bce_v(p,t)$ is computed for all uninformed nodes. Thus, calculating the weights takes $O(n^2 \cdot m)$ in total. Calculating maximum weighted matching is viewed as an external routine in [3]. Therefore, no precise algorithm for matching is introduced. The total time complexity without matching of Round_Heuristic is $O(R \cdot n^2 \cdot m)$, in which R represents the rounds of broadcasting.

There are no theoretical results for this heuristic. However, many of the testing results in several commonly used topologies are equal to the real broadcast time. By using the heuristic, the testing results in the CCC_k graph, the *Shuffle Exchange* graph, the *Butterly* graph and the *DeBruijn* graphs are presented in [3].

The quality of the heuristic depends heavily on the choice of the parameters. *Dist_Exp* is the parameter of particular importance. It determines the influence of the distance between nodes and dispersion regions. The values in the range from 0.25 to 60 are used. The precise choice of two topologies are mentioned in [3]: for the *mesh*, $Dist\_Exp = 4$, and for *butterfly* graph, $Dist\_Exp = 2$.

2.2.2 Two Heuristics

In this part, two heuristics with theoretical bounds of the broadcast problem will be introduced.

The MBT heuristic is based on the following result in graph theory [19]: The set of nodes of any graph $G = (V, E)$ can be (polynomialy) decomposed into two sets of clusters A and B, such that $\mid A \mid \leq \sqrt{n}$, the clusters in $A \cup B$ are pairwise disjoint.

The first step of the heuristic is decomposing the graph into two sets of clusters, A and B. The second step requires broadcasting in A, after which comes broadcasting from A to B. The last step involves broadcasting in B. The total broadcast time of this heuristic is bound by $3 \cdot \sqrt{n} + Diam(v) + b(v)$, in which $Diam(v)$ is the diameter of the originator v and $b(v)$ is the optimal broadcast time.

Another heuristic is presented in [23], which has no name. It is named as the *poise* heuristic in this thesis for the purposes of presentation. In [23], the *poise* of a tree T is defined as the addition of maximum degree of any node in the tree and diameter of the tree. The poise of a graph G, denoted by $P(G)$, is defined as the minimum poise of any of its spanning trees. Computing the poise of an undirected graph is NP-hard. However, there is an $O(nm \log n)$-time heuristic to compute a spanning tree of a graph on n nodes with m edges, such that the poise of this tree is within $O(\log n)$ multiplied by the poise of the graph plus an additive term of $O(log^2 n)$. The following equation is also proven in [23]: $b(G) \leq O(P(G)\frac{\log n}{\log \log n})$. Based on the above theories, the *poise* heuristic is derived. The time complexity of this heuristic is $O(nm \log^2 n)$,

and the upper bound of the broadcast time is $O(b(G)\frac{\log^2 n}{\log\log n})$.

2.3 Three Models

Ns is an event simulator targeted at networking research. It began as a variant of the REAL network simulator in 1989. The ns-2 simulator could create topologies by using several models. To compare the new heuristic with other algorithms, three different network design models from ns-2 are considered: GT-ITM *PureRandom* [26], GT-ITM *Transit – Stub* (TS) [26] and *Tiers* [8].

One purpose of designing models is to test new algorithms. When we get a new algorithm, a question frequently asked is whether it works well on real networks. If we expect to test an algorithm on a real network, a model of such a network is necessary. In some cases, a model could impact the performance of an algorithm greatly. Therefore, finding a good model is essential when valuing a new algorithm. One of the most important criteria in valuing a model is whether it could accurately model the real network. The *Tiers* model has such an advantage. It is one of the best network design models for testing networks. This model is designed to generate test networks for routing algorithms. The model produces graphs corresponding to the data communication networks such as **IP** network and **ATM** network. As mentioned in [8], several previously-known models fail to reflect one or more observations of the real networks, and Tiers is a more realistic model than these previously-known models. Rather than use the idea of a network as a non-hierarchical series of nodes, the

Tiers model considers an internetwork as a series of smaller networks and concentrates upon the relationships between these smaller networks, and the relationships between the even-smaller networks, which make up each level of a network. This follows the concept of layers in a network more closely, where each layer is built upon a number of instances of the previous layer.

WAN, MAN and LAN compose the graphs created by the Tiers model. WAN, MAN and LAN stand for the Wide Area Network, Metropolitan Area Network and Local Area Network respectively. LANs such as Ethernet and Token Rings are basically modeled as star topologies. However, LANs should be modeled as being interconnected with small degree. Each major institution (MAN) has a small number of connections to a WAN service provider. Figure 15 shows a two node WAN, with a single MAN and its five attached LANs. Each LAN has twenty-five nodes.

The graph generator TIERS created by Matthew B. Doar is used to generate random graphs. The generator is available on the Web. The major parameters chosen for this model are:

- N_w, the number of WANs; and S_W, the number of nodes in a WAN. The value of N_w is always set as 1 for the reason of simplicity.

- N_M, the number of institutional networks (MANs); and S_M, the number of nodes per MAN.

- N_L, the number of LANs per MAN; and S_L, the number of nodes per LAN.

The total number of nodes in the graph N is given by $N = S_w + N_M S_M + N_M N_L S_L$.

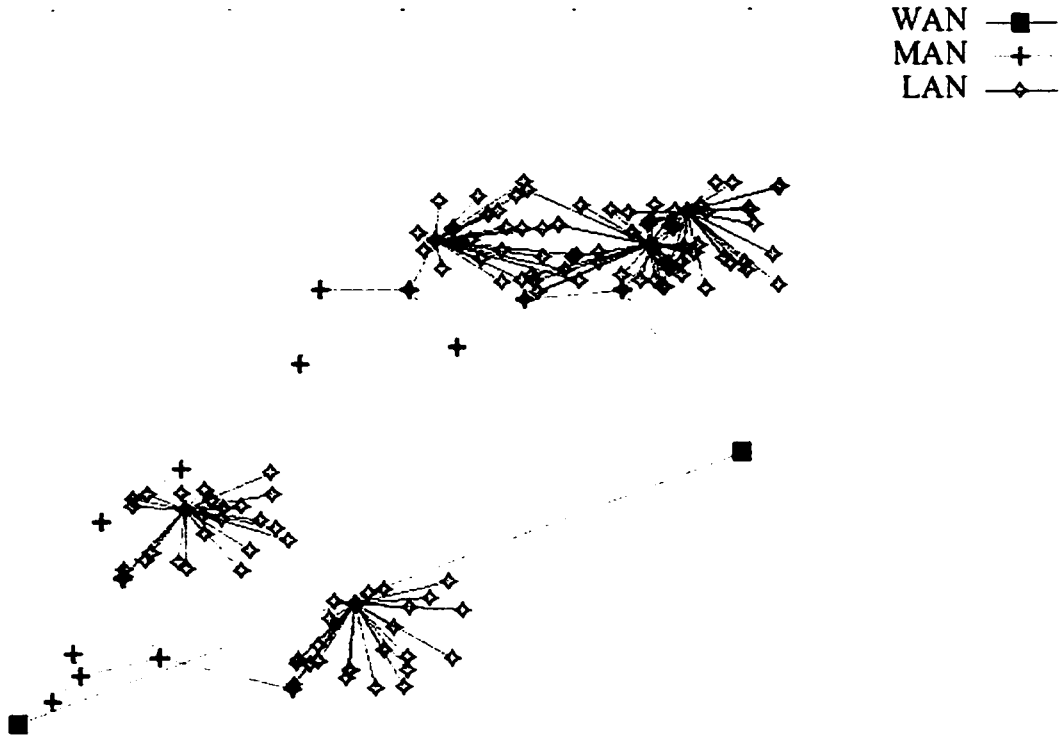


Figure 15: Generated Graph by Using the Tiers Model

The other parameters of the model are:

• The degree of intranetworking redundancy in the WAN (R_W), MAN (R_M) and LAN (R_L). This is expressed simply as the degree from a node to another node of the same type.

• The degree of internetwork redundancy between networks. This is the number of connections between a MAN and a WAN (R_{MW}), or a LAN and a MAN (R_{LM}).

Models are also helpful when study massive and dynamic networks. The Internet is such a network. There are a great number of hosts on the Internet, and thousands of new hosts join in every day. The exact structure of such a system is still unknown today. During the iterative interplay between observation and modeling, people could learn the Internet gradually. If we want to know the performance of an algorithm on the Internet, we must rely on a model. On the other hand, the testing result could also be helpful in further modeling.

GT-ITM Transit-Stub is a well-known model for the Internet. The Internet can be viewed as a set of *routing domains*. A domain is a group of hosts on the Internet. We can consider a domain to be an independent network. All nodes in a domain share routing information. There are two types of domains; one is the *stub* domain, another is the *transit* domain. When considering any path through a domain, if at least one of the end nodes is in the domain, it is a stub domain. Otherwise, the domain is a transit domain. Just like the real Internet, interconnected transit and stub domains compose the graphs generated by GT-ITM Transit-Stub. Moreover, the Transit-Stub

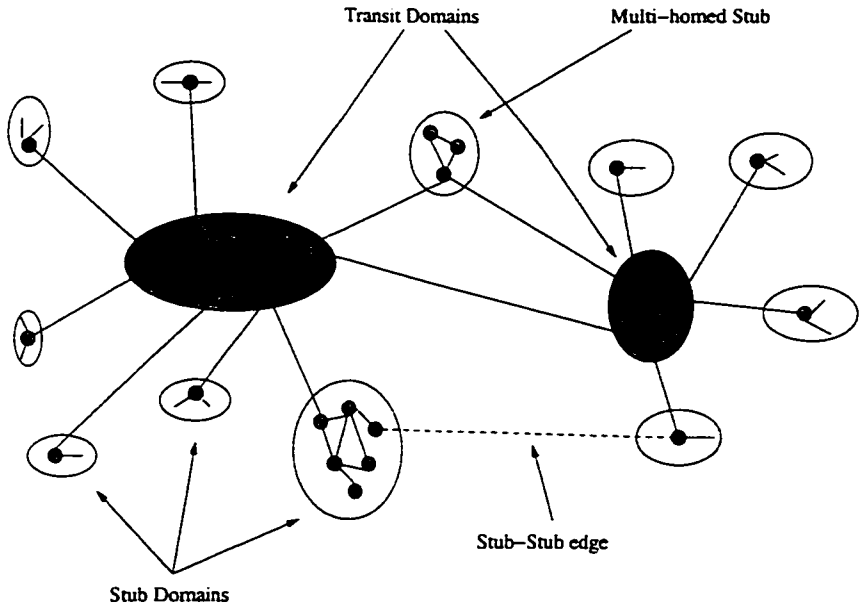


Figure 16: Example of Internet Domain Structure

model could produce graphs having realistic average node degrees.

Stub domains can be further classified as single-homed and multi-homed Stub. Multi-homed stubs connect to more than one transit domain, while Single-homed stubs connect to only one transit domain. A transit domain is composed of a set of *backbone* nodes, which are fairly highly connected to each other (a backbone node has a degree at least 2). In a transit domain, each backbone node connects to other transit domains. A stub domain has one or more *gateway* nodes, which have links to the transit domains. Some stubs also have links to other stubs. (see Figure 16)

GT-ITM PureRandom is a standard random graph model. This model places nodes in a plane at random. Considering each pair of nodes, an edge is added between them with probability p. Many models are variations of this model. This model is

often used in studying networking problems, although it does not correspond with real networks.

The graph generator GT-ITM created by Georgia Institute of Technology is used to generate graphs. The generator is available on the Web.

When using the pure random model, the most important parameter is the probability of an edge.

When using the GT-ITM Transit-Stub model, the user could employ the following parameters:

- The number of transit domains, and the number of stub domains connected to each transit node.
- The number of transit-stub and stub-stub edges.
- The number of nodes in the transit domains and the stub domains.
- The probability of an edge between each pair of nodes in the transit domains and stub domains.

Chapter 3

A New Heuristic

A new heuristic is elaborated for broadcasting in this chapter. First, an example illustrates the working process of this heuristic. Then the heuristic is formally presented in the second and third section. Finally, a refinement of this heuristic is introduced.

3.1 An Example

To explain the heuristic, we first consider an example to illustrate the process of broadcasting by using the new heuristic.

The original graph is presented in Figure 17 (a). Node a is the originator. The Figure 17 (b) gives the broadcast schedule created by the new heuristic. The numbers on the edges indicate the round in which the edge is used during broadcasting, and arrows present the direction in which the message is sent. We can see that, in this

particular graph, the new heuristic gives the optimal broadcast schedule.

Figures 18 (a) and 18 (b) illustrate the first and second rounds of broadcasting respectively. The nodes with shadowed background are informed nodes. The numbers in circles are the weights assigned to nodes. In the first round, as shown in Figure 18 (a), only a, the originator is informed. Node a has three neighbours, in which node b and node c have the same weight 5 while node d has weight 3. The area defined by the dashed line with shorter dash, which is located on the left, is $DG(c)$, and the area defined by the dashed line with longer dash, which is located on the right, is $DG(d)$. The weight of node c is 5 which means that broadcasting in $DG(c)$ that originated at node c takes 5 rounds. The possible matching for a is either b or c. In such a situation, the heuristic just selects one of them arbitrarily. So at time 1, a sends the message to c.

In the second round, as shown in Figure18 (b), nodes a and c are informed nodes. Node a has two uninformed neighbours node b and node d. The weight of node b is 0 because $DG(b)$ has only one node b.

Note that e is not a child of b in this round, because the distance between node e and the nearest informed node of e is equal to the distance between node b and the nearest informed node of b. Node c has only one uninformed neighbour node e. Thus, the matching of step two is $\{ (a, d), (c, e) \}$. Finally, the new heuristic gives the broadcasting schedule shown in Figure 17 (b).

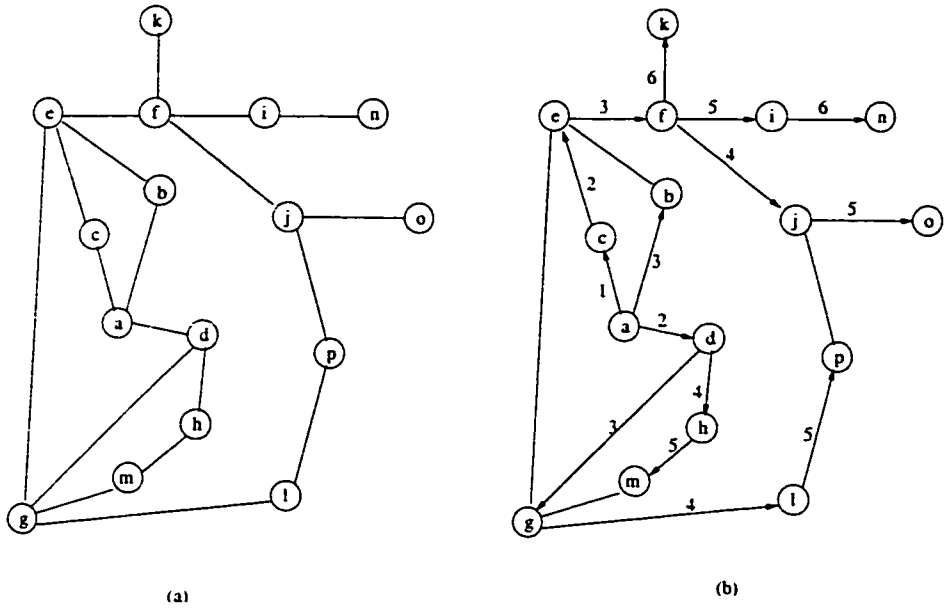


Figure 17: The Example Graph and Broadcast Schedule

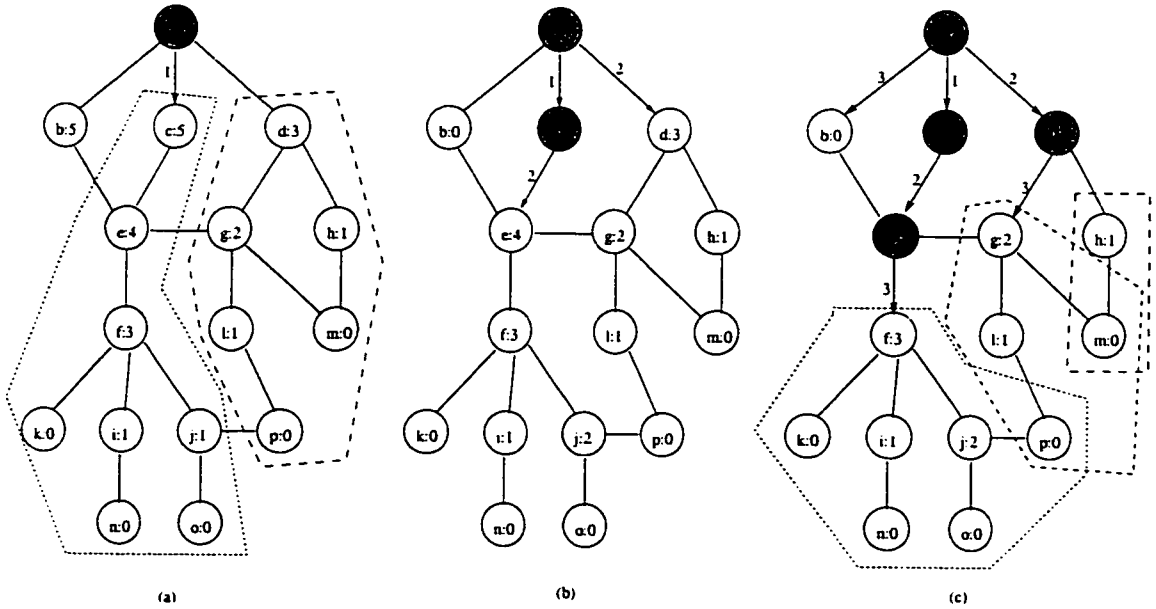


Figure 18: The Broadcasting Process in the Example Graph

It is possible that several descendant trees share nodes. We can see this in Figure 17 (c), which presents the third step of broadcasting, the areas representing $DG(f)$, $DG(g)$ and $DG(h)$ overlap.

3.2 Definitions and Implementation Algorithms

Here, the new heuristic will be formally presented.

Definition 1 *The dark region, which is denoted by DR(t), is a subset of nodes in G that is composed of all uninformed nodes at the beginning of round t. Those nodes in DR(t) that have informed neighbours, compose the dark border, which is denoted by db(t). The bright border bb(t) is composed of those informed nodes that have uninformed neighbours.*

Figure 19 illustrates how these concepts are defined. The shadowed area indicates the dark region $DR(t)$. The nodes in $DR(t)$ with the black backgrounds belong to $db(t)$, and the nodes not in $DR(t)$ with shadowed backgrounds belong to $bb(t)$.

Definition 2 For an uninformed node v, *D(v, t) is the shortest distance from node v to a node in bb(t) at round t. Given an uninformed node u and its uninformed neighbour v, if D(u, t) = D(v, t) + 1, we say u is a child of v. Node v and all its descendants make up the descendant graph of v, which is denoted by DG(V, E, v), or rather DG(v). We use a way to estimate the broadcast time of DG(v) in round t, and denote it by EB(v, t). We also use EB as the abbreviation for estimated time.*

In the new heuristic, $EB(v,t)$ is defined by the following recursive way:

(1) If at round t, a node v has no children, we define $EB(v,t) = 0$.

(2) If v has k children, and c_1, c_2, ..., c_k are the children of v ordered so that $EB(c_i,t) \geq EB(c_{i+1},t)$, then $EB(v,t) = max\{\ EB(c_i,t) + i\ \}$, for $1 \leq i \leq k$.

Here, we introduce a simple algorithm to calculate $EB(v,t)$ in $O(k)$ time given EB of all children of node v. First, find $max\{EB(c_i,t)\}$, and denote it by MAX. Then, create k buckets, and number them from 0 to $k-1$. Consider any child c, if $MAX - i \geq EB(c,t) > MAX - i - 1$, put c into the ith bucket. In fact, we need not record all the elements in each bucket; recording only the minimum value and the number of elements is enough. After that, we denote the number of elements in the first ith buckets as $SUM(i)$ and the minimum value in the ith bucket as $MIN(i)$. Then, $EB(v,t) = max\{SUM(i) + MIN(i)\}$, for $0 \leq i < k$. The proof of the last statement is presented as follows:

Proof: Given a node v has k children, and c_1, c_2, ..., c_k are the children of v ordered so that $EB(c_i,t) \leq EB(c_{i+1},t)$, then $EB(v,t) = max\{EB(c_i + i\}$, for $1 \leq i \leq k$. $EB(c_i)+i$ is named *order-weight* of c_i. Because $MAX - i \geq EB(c,t) > MAX - i - 1$ for any child c in the ith bucket, the maximum difference among EB of the children in this bucket is less than 1. Therefore, in the ith bucket, the child with the minimum EB has the maximum *order-weight*, which is equal to $SUM(i) + MIN(i)$. Thus, $max\{SUM(i) + MIN(i)\}$, for $0 \leq i < k$ is the maximum *order-weight* of all the children, which is $EB(v,t)$. □

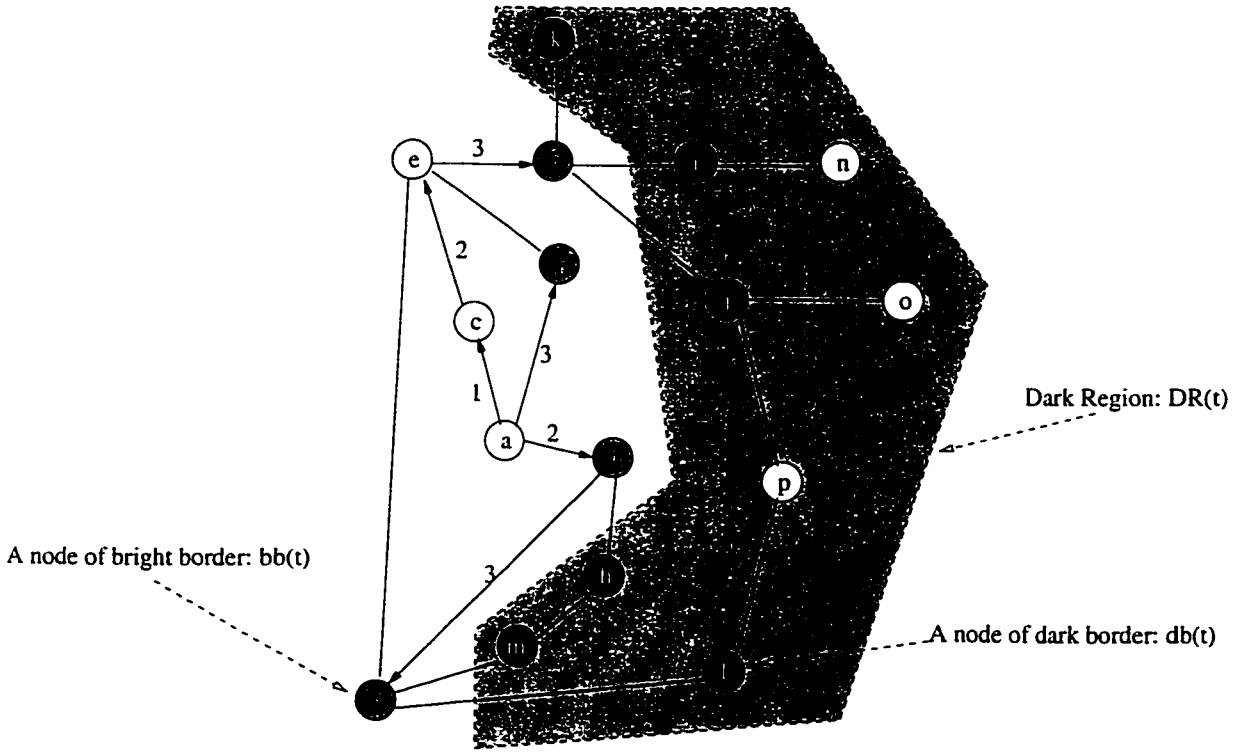


Figure 19: Several concepts in the new heuristic

Definition 3 *mnw(t) stands for the maximum-number-weight matching between set db(t) and bb(t) at round t. mnw(t) has the following properties:*

(1) Only nodes in db(t) have weight. The weight of a node in db(t) is assigned as EB of the node.

(2) The number of node pairs, where one is in db(t) and the other is in bb(t), is maximized.

(3) In case of (2), the sum of weights in db(t) is maximal.

In $mnw(t)$, the weight is assigned on nodes rather than edges. In round t, $mnw(t)$ could be calculated by the following process:

(1) In $bb(t)$, select the node v with minimum number of uninformed neighbours.

(2) Among the uninformed neighbours of v, get the node u with maximum weight. Then, u is the mate of v.

(3) Mark u as informed, then go to step 1.

By using this process, the node pairs in $mnw(t)$ must be at a maximum. However, the process would not always make the weights in $db(t)$ reach their maximum. In fact, this process is a heuristic for calculating $mnw(t)$.

Two algorithms are defined to implement the above process. Both of them are *greedy* algorithms. They are named *sort-matching* and *lists-matching* respectively. The time complexity and space complexity of these two algorithms are presented in Table 2. We can see that these two algorithms are a trade off between time and

	time complexity	space complexity
sort-matching	$O(n^2)$	$O(n)$
lists-matching	$O(m)$	$O(\Delta \cdot n)$

Table 2: Time and Space Complexity of Matching

space. In other chapters of this thesis, we only consider the *lists-matching*, which has a lower time complexity. Users who prefer a small space complexity could choose *sort-matching*.

The *sort-matching* algorithm is a straightforward algorithm derived from the above process. In this algorithm, the set $bb(t)$ is saved in a list. The match is found as follows:

(1) Find the node $v \in bb(t)$ where v has minimum number of uninformed neighbours.

(2) Among the uninformed neighbours of v, get the node u with maximum weight. Then, u is the mate of v.

(3) Mark u as informed, then go to step 1.

We assume that $bb(t)$ contains k nodes and the degree of the ith node is $d_i, 0 \leq i \leq k-1$. Finding a node with a minimum number of uninformed neighbours takes k units of time. For the ith node, finding the uninformed neighbour with maximum weight takes d_i units of time. Then, finding the mate for the ith node takes $k+d_i$ units of time in total. Therefore, finding the mates for k nodes takes $k^2 + \sum_{i=0}^{i=k-1} d_i$ units of time. The maximum possible value of k is n, which is the number of nodes in the

network. So, the maximum needed time is $n^2+\sum_{i=0}^{i=n-1} d_i$. Because $\sum_{i=0}^{i=n-1} d_i = 2m$, in which m is the number of links in the network, the time complexity of *sort−matching* is $O(n^2+m) = O(n^2)$. The set $bb(t)$ is saved in a list. The maximum size of this list is n. Thus, the space complexity of *sort-matching* is n.

In *lists-matching* algorithm, the set $bb(t)$ is saved as follows:

```
Class BrightBorder{
        List * bb[iDegree];
        ListNode ** PositionList[n];
        }
```

The *iDegree* is the maximum degree, and n is the number of nodes in the graph. A node in $bb(t)$ with i uninformed adjacent nodes is saved in list $bb[i]$. In each list $bb[i]$, there is a *current* pointer that indicates the current position, and the *tail* pointer that indicates the last element in the list. Initially, the *current* points to the *head* of list $bb[i]$. We say a list $bb[i]$ is matched if the *current* of this list is equal to the *tail* of this list. Otherwise, the list is unmatched. The matching is done when all the lists are matched. When doing matching, in each step, a node v that lies after the *current* of the first unmatched list $bb[i]$, for $1 \leq i \leq iDegree$, is taken. Then, from all the uninformed neighbours of v, the one with maximum weight is selected as the mate of v. After that, u is marked as informed. The pointer *current* moves to point the element after v. Since u is informed, the number of uninformed neighbours of all u's neighbours should be decreased by one. Also, if such a node is in $bb[i]$, it should

be moved to $bb[i-1]$. Such a node is saved before or after the *current* of $bb[i-1]$ depending on whether this node is matched or not. The list *PositionList* records the entry of all nodes of $bb(t)$ in lists $bb[i]$, where $1 \leq i \leq k$. So, searching for a node before moving it is not needed.

Every node in $bb(t)$ must be visited to do matching in each round. When a node is matched, all its neighbours have to deduct their number of uninformed neighbours by one. This can be done in $\sum_{i=1}^{n} O(d_i)$ time, where d_1, d_2 ..., d_n are the degrees of the nodes of the graph. Since $\sum_{i=1}^{n} d_i = 2m$, then the complexity of $lists - matching$ will be $O(m)$.

The class used to save $bb(t)$ takes $\Delta \cdot n$ units of space. So, the space complexity of $lists - matching$ is $O(\Delta \cdot n)$.

3.3 The Heuristic and its Complexity

Based on the above definitions, the heuristic is defined as follows:

Algorithm:

(1) Initialize $bb(t)$ so that $bb(0)$ has one node: the originator.

(2) Put $EB(v,t)$ as the weight to any node v in $DR(t)$.

(3) Find the $mnw(t)$ between $bb(t)$ and $db(t)$, and during the process, mark all matched nodes as informed.

(4) Compute $bb(t+1)$.

	time complexity	space complexity
using sort-matching	$O(R \cdot n^2)$	$O(m)$
using lists-matching	$O(R \cdot m)$	$O(\Delta \cdot n)$

Table 3: Time and Space Complexity of the Heuristic

(5) If $bb(t+1)$ is empty, the process is complete, and t would be the broadcast time. Otherwise, go to (2).

Steps 2 and 3 dominate the time complexity. Step 2 (the implementation of assigning weights) has two phases. In the first phase, the heuristic performs a Breadth First Search (BFS) of $DR(t)$ from the $bb(t)$ and label each node by $D(v,t)$. At the same time, a set is created to save nodes that have no *children*. This set is denoted by $rb(t)$, which means remote border. Let m denote the number of edges of graph G, then this step can be done in $O(m)$ time. In the second phase, a recursive process is used to compute weight for each node in $DR(t)$. This process starts from $rb(t)$ towards $db(t)$. In the worst case, we have to calculate $EB(v,t)$ for every node of graph G. The degree of the ith node of graph G is denoted by d_i. By using the new heuristic to calculate $EB(v,t)$, the time needed for a node with degree d is $O(d)$. Then, the time needed to calculate all the nodes is $\sum_{i=1}^{n} O(d_i)$. Since $\sum_{i=1}^{n} d_i = 2m$, the complexity of this step is $O(m)$. Thus, the total complexity of assigning weights (step 2) is $O(m)$. When using the $sort - matching$ algorithm in step 3, the time complexity in one round is $O(n^2) + O(m) = O(n^2)$. When using the $lists - matching$ algorithm in the step 3, the time complexity in one round is $O(m) + O(m) = O(m)$. During the process of matching in round t, whenever a node is marked as informed,

it is added into $bb(t)$. At the beginning of round $t+1$, each node in $bb(t+1)$ will be visited once. If the number of uninformed neighbours of a node in $bb(t+1)$ is 0, then this node will be removed from $bb(t+1)$. In the worst case, this take $O(n)$ time. Let R denote the rounds of broadcasting; the total time complexity of the new heuristic for broadcasting is $O(R \cdot n^2)$ or $O(R \cdot m)$.

When using the $sort-matching$ algorithm, saving the graph dominates the space complexity, which is $O(m)$. When using the $lists-matching$ algorithm, the matching dominates the space complexity, which is $O(\Delta \cdot n)$. The time and space complexities in different situations are presented in Table 3.

3.4 Refinement

Because a child could have more than one parent, the weight of such a child could be counted several times when we compute the weight of those parents. Thus, the effect of this child during the process of broadcasting is overestimated. Figure 20 shows an example. The left figure in Figure 20 is the original graph. Node a is the originator. The right figure in Figure 20 gives the broadcast schedule by using the new heuristic in this example. The middle figure in Figure 20 presents the weights of each uninformed node in round 2. The nodes with shadowed backgrounds are informed nodes. As mentioned before, the weight of a node is the approximate time before broadcasting is finished after the node is informed. By using the new heuristic, in the second round, the weight of node f and c are both 1 because node g is a child

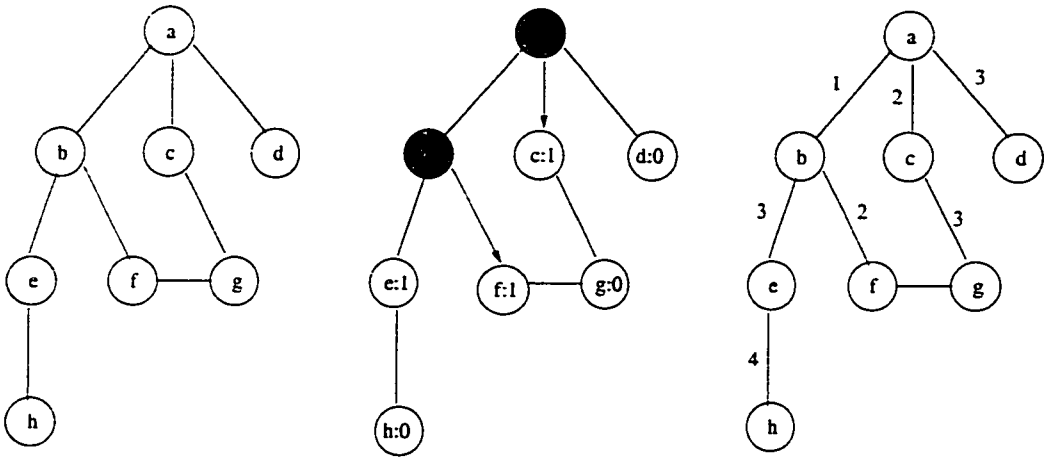


Figure 20: An Example Graph

of them. However, node g need not be informed by both node f and c. Either node f or c sending the message to node g will let g be informed. We can see that the attribution of node g in assigning weight is overestimated. Because node e and node f have the same weight, node b could send the message to node f in the second round, although sending to node e is a better choice. This motivates the following refinement, dividing the weight of this child by the number of its parents.

Formally, we can calculate $EB(v,t)$ in the following way:

(1) If at round t, a node v has no *child*, then $EB(v,t) = 1$.

(2) Given v has k *children*. Those *children* are denoted by c_i, $1 \leq i \leq k$. Assume further that $EB(c_i,t) \geq EB(c_{i+1},t)$, then $EB(v,t) = max\{ \frac{EB(c_i,t)\cdot p}{n} + i \}$, in which n is the number of parents of c_i.

The left figure in Figure 21 presents the weights of each uninformed node in round

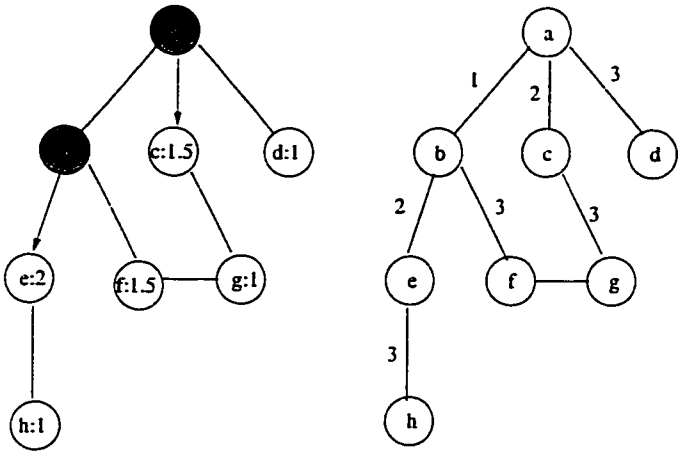


Figure 21: The Performance of Refinement

2 by using the refinement. The right figure in Figure 21 shows the broadcast schedule by using the refinment. We can see that this is the optimal broadcast schedule of the example graph.

In the above statement, p is a parameter. We can simply assign $p = 1$ to all kinds of topologies. In such a situation, the performance is improved in some topologies. But, the improvement is not significant. If suitable values of the parameter are used in a special topology, the performance could be improved more. Giving values 2, 3 and 4 to parameters p for the *Cube-Connected Cycle*, the *ButterFly* and the *ShuffleExchange* graph respectively, some of the results could be imporved.

Note that the time complexity and space complexity of the refinement are the same as the time complexity and space complexity of the original heuristic.

Chapter 4

Theoretical Results

By using the new heuristic, the broadcast time is equal to the optimal broadcast time in several topologies. As mentioned in the introduction, when given a node u in graph G, $b(u, G)$ denotes the optimal broadcast time in G when u is the originator. $b(G)$ stands for the optimal broadcast time in G. The number of rounds necessary to complete broadcasting in graph G by using a heuristic A is denoted by $b(A(G))$. Also, $b(A(u, G))$ represents the broadcast time in G by using heuristic A when the originator is node u.

In Section 4.1 and Section 4.2, it is proven that, in several topologies, the new heuristic generates the same broadcast time as the optimal broadcast time, which is presented in Chapter 2. In Section 4.3, it is proven that the new heuristic performs as the same as the optimal broadcast scheme in tree.

4.1 Several Simple Topologies

• *The complete graph K_N:* By using the new heuristic in each round, the *weight* of each uninformed node is 0. This happens because the distance between any uninformed node to the $bb(t)$ (the bright border, which is introduced in Chapter 3) is 1. Thus, for each uninformed node u, the $DG(u)$ (the descendant graph of u, mentioned in Chapter 3) contains only one node u. Then, the weight of u equals 0. In such a situation, every informed node sends the message to an arbitrary uninformed neighbour. In this case, the weight of nodes does not apply. However, because the complete graph is such a good topology for broadcasting, the new heuristic gives optimal broadcast time in any complete graph.

Theorem 4.1 $b(A(K_N)) = \lceil log_2 n \rceil = b(K_N)$.

Proof: By using the new heuristic, in a round k, an informed node must send the message to an uninformed one if there are indeed uninformed nodes in the end of this round. Since every node in K_N is linked to all other nodes, every informed node has uninformed neighbours in the end of a round k unless all nodes in G are informed. Thus, the informed nodes must be doubled in each round except in the last round of broadcasting. Therefore, by using the new heuristic, the broadcast time in any complete graph K_N is $\lceil log_2 N \rceil$, which is equal to the optimal broadcast time in the complete graph. □

Figure 22 presents the weights of nodes in K_6 in the first round. The left figure is the original graph. The node with the black background is the originator. In the

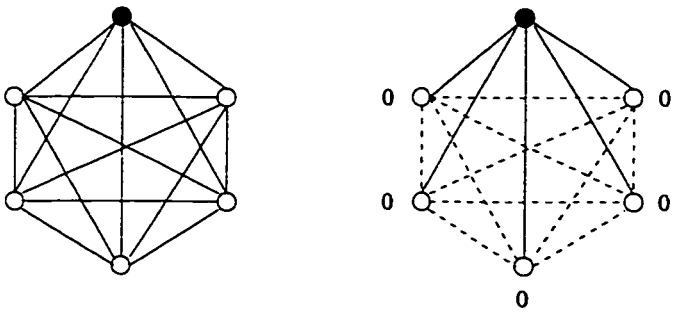


Figure 22: Weights of Nodes in K_6

right figure. the solid lines form the BFS tree of K_6.

- *The path graph P_N:* During the process of broadcasting by using the new heuristic, the informed nodes separate P_N into two sub-uninformed areas. In each area, the weight of an uninformed node v is the distance between v and one end of P_N in the area. In the first round, we assume that the distance of the node u to the two ends E_1 and E_2 of P_N are l_1 and l_2, and $l_1 \geq l_2$. Thus, the weight of two neighbours of u are $l_1 - 1$ and $l_2 - 1$ respectively. When l_1 is greater than l_2, the originator must send the message in the direction to the end E_1. In the later steps, the two nodes that have an uninformed neighbour send the message to their own neighbours. This takes l_1 round in total. In such a case, $l_1 \geq l_2 + 1$. When $l_1 = l_2$, the weight of the two neighbours are equal. In this situation, the new heuristic selects one neighbour arbitrarily to send the message in the first round. In the later steps, two nodes are informed in all rounds, except in the last. This takes $l_2 + 1$ rounds in total. In this case, $l_1 < l_2 + 1$. In summary, $b(A(u, P_N)) = max\{l_1, l_2 + 1\}$, $l_1 \geq l_2 = b(u, P_N)$. When u is an end of P_N, $b(A(P_N)) = n - 1 = b(P_N)$. In conclusion, we have the

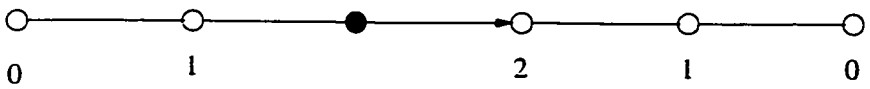


Figure 23: Weights of Nodes in P_6

following theorem:

Theorem 4.2: $b(A(P_N)) = n - 1 = b(P_N)$.

Figure 23 presents the weights of nodes in P_6 in the first round. The node with the black background is the originator. The arrow stands for the direction in which the message is sent in the first round.

- *The ring graph C_N:* No matter what the weights of the adjacent nodes of the originator are, the originator sends the message to one neighbour in the first round. Then, based on the definition of *mnw* (the maximum-number-weight matching, see Chapter 3), the two informed nodes with uninformed neighbours will send the message to their respective neighbours in each round. This scheme is the same as the optimal broadcast scheme (introduced in Chapter 2) in the ring graph. Therefore, we may draw the conclusion that:

Theorem 4.3 $b(A(C_N)) = b(C_N)$.

Figure 24 illustrates the two possible broadcast schemes by using the new heuristic in C_6. The node with the black background is the originator, and the number adjacent to an edge stands for the round in which the edge is used.

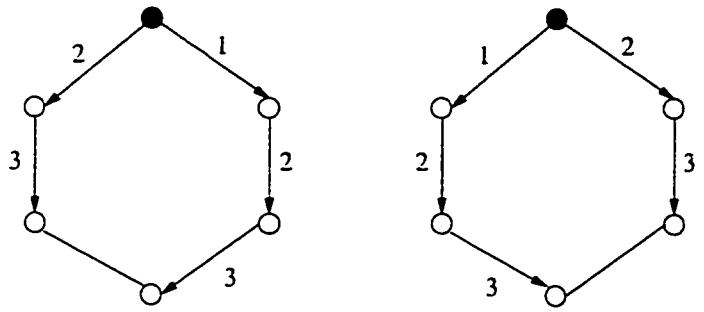


Figure 24: Two Possible Broadcast Scheme in C_6

4.2 The Grid Graph

In this section, the following theorem is proven:

Theorem 4.4 In a grid graph, when the originator is in the corner, $b(A(G_{m,n})) = m + n - 2 = b(G_{m,n})$.

To prove theorem 4.4, a simple algorithm for broadcasting are defined: in each round, any informed node sends the message to one of its uninformed neighbours if such a neighbour exists. In this section, this simple algorithm is denoted by S, and the broadcast time in a graph G by using this algorithm is denoted by $b(S(G))$. The statement $b(S(G_{m,n})) = m+n-2 = b(G_{m,n})$ will be proven. In the new heuristic, if a node has any uninformed neighbours in a round, then the node will have a mate. As a result, the node will always send the message to one of its neighbours. Therefore, the new heuristic is also an S algorithm.

A node in the grid graph is denoted by $N_{i,j}$, in which i and j are the row and column of the grid, respectively. To present the proof, we need the following definitions:

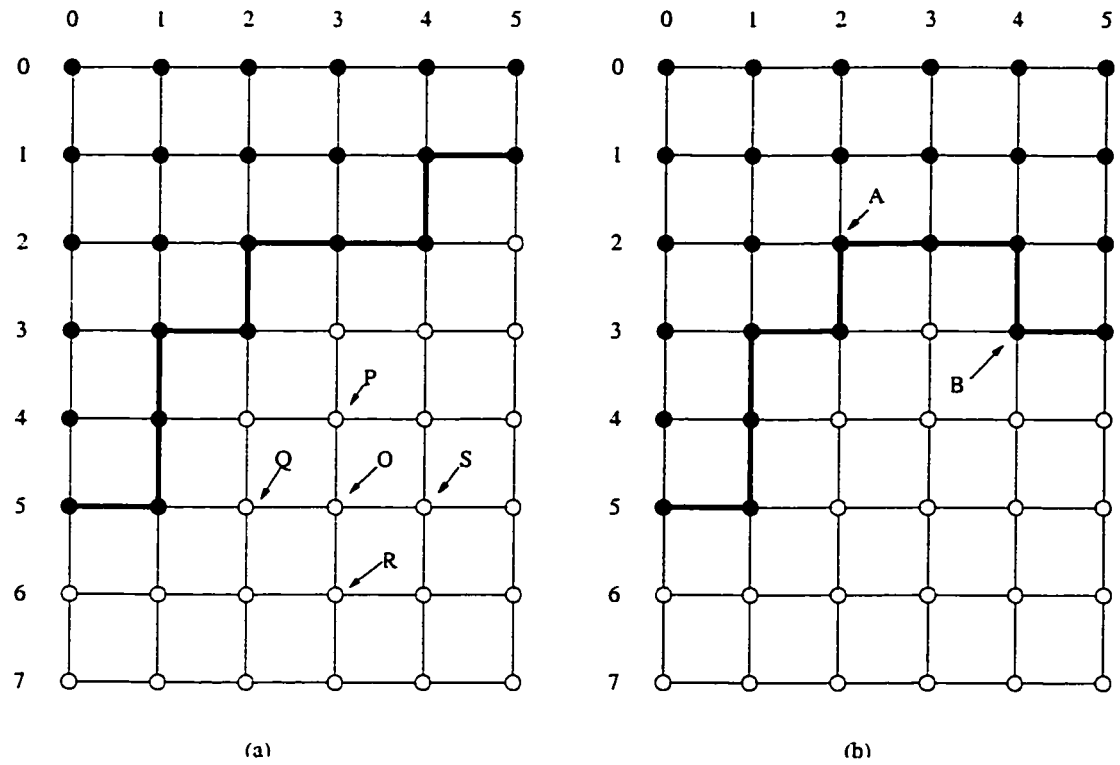


Figure 25: Definitions in the Grid Graph

(1)*border*: the set of nodes that have uninformed neighbours.

(2)*inside neighbours* of node $N_{i,j}$: $N_{i-1,j}$ and $N_{i,j-1}$

(3)*outside neighbours* of node $N_{i,j}$: $N_{i+1,j}$ and $N_{i,j+1}$

(4)*outside situation*: A status under which there are no two such nodes on the border wherein both the values of the row and the column of one node are respectively greater than those from another node.

Figure 25 illustrates the above definitions. The nodes with black backgrounds are informed nodes, and the nodes with white backgrounds are uninformed nodes. The

nodes connected by bold edges compose the *border*. Considering the node O, node P and Q are the two inside neighbours of node O, and nodes R and S are the two outside neighbours of node O. The left graph in Figure 25 applies to the outside situation, while the right graph does not. We can see that in the right graph, the row and column of node B are 3 and 4 respectively, which are both greater than the respective value of node A (2 and 2 respectively).

Lemma 4.1 Under the outside situation, any node on the border has no inside uninformed neighbour.

Proof: Consider a node $N_{i,j}$ on the border, assume that $N_{i,j}$ has an inside neighbour M. Then $N_{i-1,j-1}$ is also on the border because it has at least one uninformed neighbour M. So, both $N_{i,j}$ and $N_{i-1,j-1}$ are on the border, and this is contradictory to the definition of the outside situation. □

Lemma 4.2 Starting from an outside situation, after every node on the border sends a message to one of its uninformed neighbours, the new border is still applied to the outside situation.

proof: When considering any two nodes $N_{i,j}$ and $M_{p,q}$ on the border, there are only three possibilities: $i \neq p$ and $j \neq q$, $i \neq p$ and $j = q$ and $i = p$ and $j \neq q$.

Considering the first possibility, we assume $i > p$ and $j < q$. Let the node informed by N and M in the new border be N' and M' respectively. Because of lemma 3.1, N' could only be $N'_{i+1,j}$ or $N'_{i,j+1}$, and M' could only be $M'_{p+1,q}$ or $M'_{p,q+1}$. Obviously, it is impossible that both the value row and column of N' are greater than that of M',

and vice versa.

Under the second condition, $i \neq p$ and $j = q$. We assume that $i > p$ and $j = q$. Then, N and M can be denoted by $N_{i,j}$ and $M_{p,j}$.

We can see that $M'_{p+1,j}$ is not an uninformed neighbour of M, because if $p+1$ is equal to i, then M' equals N. Otherwise, if $p+1 < i$, and $M'_{p+1,j}$ is a neighbour of M, then there is a node $Q_{p+1,j-1}$ which has an uninformed neighbour M'. Therefore Q is on the border. However, this is a contradiction to the outside situation if we consider N and Q, because $i > p+1$ and $j > j-1$.

Therefore, M has only one uninformed neighbour $M'_{p,j+1}$. N has two possible uninformed neighbours $N'_{i+1,j}$ or $N'_{i,j+1}$. Clearly, M' and N' apply to the outside situation.

The proof of the third possibility is the same as that of the second one.

Then, lemma 4.2 is proven.□

Lemma 4.3: Under the outside condition, if node $N_{i,j}$ is informed, then there are no such nodes $M_{p,q}$ ($0 \leq p \leq i$ and $0 \leq q \leq j$) that are not informed.

proof: If $p = i$ and $q = j$, then $M = N$. Therefore, M is informed. Now we consider the following two situations: $p < i$ or $q < j$. Assume M is uninformed, then both $P_{p-1,q}$ and $Q_{p,q-1}$ are nodes on the border. When $p < i$, the values of both row and column of Q are less than those of N. This is contradictory to the outside condition. When $q < j$, the values of both row and column of P are less than N values. Again, this is contradictory to the outside condition. Therefore, M must

have been informed.□

Now we can begin to prove theorem 4.4 . In the first step of broadcasting in $G_{m,n}$, only $N_{0,0}$ is informed and the *border* only has one node N. Obviously, the border is under the outside situation. From lemma 4.1, we know that in each round, the nodes on the border can only send the message to their outside neighbours. Therefore, the longest distance between the border and the originator increases by 1 in each step. From lemma 4.2, the outside situation has never been changed throughout the whole process of broadcasting. From 4.3, we can see that the node $N_{m,n}$ must be informed in the last step, because every node $M_{p,q}$ in the grid has a row less than or equal to m, as well as a column less than or equal to n. Therefore, the node $N_{m,n}$ can be informed at step $m+n-2$ because the length of the path between the originator and $N_{m,n}$ is $m+n-2$. After $m+n-2$ steps, the broadcasting is completed.

4.3 Tree

The *broadcast center* of G, $BC(G)$, is the set of all nodes that have a minimum broadcast number: $BC(G) = \{u \mid b(u) = b(G)\}$. In [25], an $O(N)$ algorithm for determining the broadcast center of any tree with N nodes was presented. A by-product of this algorithm is $b(u,T)$ of any node u in a tree T.

If (u,v) is an edge in tree T, then deleting (u,v) from T could produce two subtrees of T. One such subtree containing v is denoted by $T(v,u)$. Let $v_1, v_2, \cdots, v_k$ denote the nodes adjacent to u in T, and assume they are labeled so that $b(v_1, T(v_1,u)) \geq$

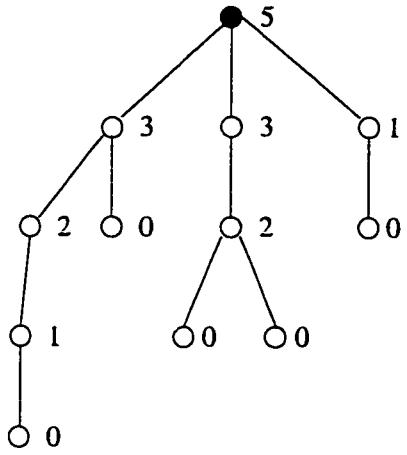


Figure 26: Roadcast Time of Tree

$b(v_1, T(v_2, u)) \geq \cdots \geq b(v_k, T(v_k, u))$. Since $b(v_i, T(v_i, u))$ is the amount of time it will take to pass the message from v_i to the other nodes in $T(v_i, u)$ after a call from u to v_i, the optimal calling sequence from u is to first call v_1, then v_2, then v_3, etc.

Given the originator u in a tree T, we can determine $b(u, T)$ by using the following recursive process. $BT(v)$ stands for the amount of time needed to inform all children of a node v in T after the node v is informed. Let the originator u be the root. $BT(v) = 0$ when v is a *leaf*. When v is not a leaf, if v has k children $v_1, v_2, \cdots, v_k$ and $BT(v_1) \geq BT(v_2) \geq \cdots \geq BT(v_k)$, then $BT(v) = max\{BT(v_i) + 1,\ 1 \leq i \leq k\}$. The $BT(v)$ of each node v in a tree is illustrated in Figure 26. The number beside a node v refers to $BT(v)$, and is also the weight of this node in the new heuristic.

Theorem 4.5 $b(A(u, T)) = b(u, T)$, for any tree T.

proof: By using the new heuristic A, a breadth first search (BFS) tree is created.

A tree has only one spanning tree, which is itself. Therefore, the BFS tree formed by using A is actually the tree T itself. From the heuristic and the above definition of BT, we can see that the weight given to a node v is equal to $BT(v)$ in T. By using A, an informed node will first send the message to its uninformed nodes with a maximum weight. This is the same scheme as the optimal broadcast scheme in a tree. □

Chapter 5

Experimental Results

In this chapter, the testing results in several commonly used topologies and three graph models are provided. The testing results in the commonly used topologies are introduced in Section 5.1, and the testing results in three well-known graph models are provided in Section 5.2.

5.1 Testing Rsults in Commonly Used Topologies

In this section, S_n, BF_d, H_d, SE_d and CCC_d abbreviate the *Star* graph, the *Butterfly* graph, the *Hyper Cube* graph, the *Shuffle Exchange* graph and the *Cube-Connected Cycle* graph with dimension d, in the order stated. The *deBruijn* graph $UB(2,d)$ is obtained by removing the edge orientation in the *deBruijn* digraph $B(2,d)$ with indegree and out outdegree 2 and diameter d. In the tables presented in this section, the *Lower* and *Upper* stand for lower bound and upper bound, respectively. *Average*,

Minimum and *Maximum* translate to the average, minimum and maximum testing results. *Optimal* refers to the optimum broadcasting time of a graph.

5.1.1 The Testing Results of the Heuristic

The broadcast time obtained by using the new heuristic in the Hypercube graph H_d at most exceeds the optimal broadcast time by one. When $d \leq 7$, the minimum testing result is equal to the optimal broadcast time. Moreover, when $d \leq 4$, the heuristic always provides optimal broadcast time. Considering the testing result, when $8 \leq d \leq 20$, the new heuristic always produces a broadcast time that is $b(H_d)+1$ (see Table 4).

The lower bounds of the de Bruijn graph are not real lower bounds. They are computed with the formulas in [18], which only hold asymptotically. Thus, it may happen that for $UB(2,4)$ and $UB(2,7)$, the heuristic requires fewer rounds than are given by the lower bound.

In Table 5, we can see that the maximum testing results never exceed the upper bounds. Moreover, within the range of lower bound and upper bound, the maximum testing results are close to the lower bounds. When $d \leq 10$, the minimum testing results are equal to or less than the lower bounds, and when $11 \leq d \leq 20$, the minimum testing results exceed the lower bounds by one or two rounds when they are not equal to the lower bounds. The greatest difference between the maximum and minimum testing results is 1.

	HyperCube H_d			
D	**Optimal**	**Minimum**	**Maximum**	**Average**
3	3	3	3	3
4	4	4	4	4
5	5	5	6	5.068
6	6	6	7	6.444
7	7	7	8	7.951
8	8	9	9	9
9	9	10	10	10
10	10	11	11	11
11	11	12	12	12
12	12	13	13	13
13	13	14	14	14
14	14	15	15	15
15	15	16	16	16
16	16	17	17	17
17	17	18	18	18
18	18	19	19	19
19	19	20	20	20
20	20	21	21	21

Table 4: Testing Results in H_d

	Undirected de Bruijn $UB(2, D)$				
D	**Lower**	**Upper**	**Minimum**	**Maximum**	**Average**
3	4	6	4	4	4
4	6	8	5	5	5
5	7	9	6	7	6.334
6	8	11	8	9	8.016
7	10	12	9	10	9.566
8	11	14	11	12	11.002
9	12	15	12	13	12.366
10	14	17	14	15	14.01
11	15	18	15	16	15.53
12	16	20	17	17	17
13	18	21	18	19	18.57
14	19	23	20	20	20
15	20	24	21	22	21.82
16	22	26	23	23	23
17	23	27	25	25	25
18	24	29	26	26	26
19	26	30	28	28	28
20	27	32	29	29	29

Table 5: Testing Results in $UB(2, D)$

	Star S_n				
n	**Lower**	**Upper**	**Minimum**	**Maximum**	**Average**
3	3	11	3	3	3
4	5	14	5	6	5.951
5	7	19	8	9	8.022
6	9	23	11	11	11
7	13	29	14	14	14
8	16	33	16	17	16.9
9	19	39	20	20	20

Table 6: Testing Results in S_n

The star graphs with degrees ranging from one to nine were tested. A star graph with the degree equal to nine has $9! = 362880$ nodes. The lower bound of a star graph on N nodes is $\lceil \log N \rceil$, which is also the lower bound of any graph on N nodes. The formula in [6] are used to calculate the upper bounds of the star graph. In Table 6, we can see that the upper bounds are much greater than the lower bounds. However, the testing results are close to the lower bounds. In some cases, the maximum testing results are one round greater than the minimum testing results, and in other cases, they are equal to each other (see Table 6).

In the shuffle exchange graph, when $d = 9$, the minimum testing result begins to exceed the optimum broadcast time, and when $d = 20$, the minimum testing results exceed the optimum broadcast time by three rounds (see Table 7). We can see that the new heuristic does not perform so well as in other tested graphs. However, the refinement of the heuristic makes up for this drawback. The testing results of refinement will be presented later.

	SE_d			
d	**Optimal**	**Minimum**	**Maximum**	**Average**
3	5	5	5	5
4	7	7	7	7
5	9	9	10	9.735
6	11	11	12	11.512
7	13	13	15	14.01
8	15	15	17	16.236
9	17	18	19	18.533
10	19	20	22	20.71
11	21	22	24	23.04
12	23	24	26	25.32
13	25	27	29	27.45
14	27	29	31	29.72
15	29	31	32	31.7
16	31	33	35	34.2
17	33	36	36	36
18	35	37	38	37.8
19	37	40	42	40.6
20	39	42	42	42

Table 7: Testing Results in SE_d

	BF_d				
d	**Lower**	**Upper**	**Minimum**	**Maximum**	**Average**
3	5	5	5	5	5
4	7	7	7	8	7.034
5	8	9	9	10	9.001
6	10	11	10	11	10.997
7	11	13	12	13	12.762
8	13	15	14	15	14.316
9	15	17	16	17	16.01
10	16	19	18	18	18
11	18	21	19	20	19.99
12	19	23	21	22	21.6
13	21	25	23	24	23.15
14	23	27	25	25	25
15	24	29	27	27	27
16	26	31	29	29	29

Table 8: Testing Results in BF_d

In the butterfly graphs, the minimum testing results become greater than the lower bounds when $d = 7$. When $d = 16$, the minimum testing results are three rounds greater than the lower bounds. However, the testing results are never greater than the upper bounds (see Table 8).

For the CCC_d graph, the minimum testing results do not exceed the lower bounds until $d = 16$. Moreover, the testing results are never greater than the upper bounds (see Table 9).

5.1.2 The Testing Results of the Refinement

All six types of the graphs (S_n, BF_d, H_d, SE_d, $UB(2,d)$ and CCC_d) were tested by using the refinement of the heuristic. Among them, in H_d, S_n and $UB(2,d)$, the

	CCC_d				
d	**Lower**	**Upper**	**Minimum**	**Maximum**	
3	6	7	6	6	6
4	9	9	9	9	9
5	11	12	11	11	11
6	13	14	13	14	13.953
7	16	17	16	17	16.17
8	18	19	18	19	18.89
9	21	22	21	22	21.2
10	23	24	23	24	23.84
11	26	27	26	27	26.15
12	28	29	28	29	28.75
13	31	32	31	32	31.6
14	33	34	33	34	33.6
15	36	37	36	37	36.8
16	38	39	39	39	39

Table 9: Testing Results in CCC_d

refinement performs almost the same way as in the original heuristic. Therefore, the testing results of these three graphs have not been listed in this section.

In the butterfly graph, when $d = 5$, the value of the maximum testing results are improved by using the refinement. This is indicated by *. However, by using the refinement, when $d = 6$, the values of minimum testing results are greater than those of the original heuristic. This is indicated this by $^-$. The advantage of the refinement is that it improves the average testing results in most cases (see Table 10).

In the CCC_d graph, the refinement improves many values of the maximum (indicated by the bold print). Whenever this happens, the maximum testing results are equal to the minimum testing results, which means that the testing results of the refinement is always equal to the lower bounds. However, the refinement never improves

	BF_d		Refinement		
d	**Lower**	**Upper**	**Minimum**	**Maximum**	**Average**
3	5	5	5	5	5
4	7	7	7	8	7.006
5	8	9	9	9*	9
6	10	11	11⁻	11	11
7	11	13	12	13	12.316
8	13	15	14	15	14.008
9	15	17	16	17	16.105
10	16	19	18	18	18
11	18	21	19	20	19.86
12	19	23	21	22	21.18
13	21	25	23	24	23.15
14	23	27	25	25	25
15	24	29	27	27	27
16	26	31	29	29	29

Table 10: Testing Results in BF_d by Using Refinement

the values of the minimum. Furthermore, there are two places where the minimum testing results of the original heuristic are less, which means they are better, than those of refinement. The two places are indicated by using italic print. Generally speaking, refinement improves the average performance in CCC_d, however, it never improves the minimum testing results (see Table 11).

By using refinement, the performance in the shuffle exchange graphs is greatly improved. Especially when the d is greater than eight, the refinement improves not only the average performance but also the minimum testing results. The improvements in the minimum and maximum testing results are marked by the bold print (see Table 12).

	CCC_d		Refinement		
d	**Lower**	**Upper**	**Minimum**	**Maximum**	**Average**
3	6	7	6	6	6
4	9	9	9	9	9
5	11	12	11	11	11
6	13	14	13	**13**	13
7	16	17	16	**16**	16
8	18	19	18	**18**	18
9	21	22	21	**21**	21
10	23	24	23	24	23.27
11	26	27	26	**26**	26
12	28	29	*29*	29	29
13	31	32	31	**31**	31
14	33	34	*34*	34	34
15	36	37	36	**36**	36
16	38	39	39	39	39

Table 11: Testing Results in CCC_d by Using Refinement

5.2 Testing Results in Three Graph Models

In total, we considered roughly two hundred different graphs generated by the three models from ns-2 introduced in Chapter 2 on N (in the range from 155 to 4400) nodes. Part of the experimental results are listed in the following tables. The testing result of both the original heuristic and the refinement are provided. They are respectively presented in the column *Origin* and *Refinement*. The heuristic were tested for one hundred times in each graphs. Min, Max and Ave refer to the minimum, maximum and average testing results.

Tables 13 and 14 present the testing results in the GT-ITM pure random model. In these figures, p stands for the probability of the existence of an edge between each pair of nodes, and P represents the parameter used in the refinement. The number

	SE_d	Refinement		
d	**Optimal**	**Minimum**	**Maximum**	**Average**
3	5	5	5	5
4	7	7	7	7
5	9	9	**9**	9
6	11	11	**11**	11
7	13	13	**13**	13
8	15	15	**16**	15.502
9	17	**17**	**18**	17.469
10	19	**19**	**20**	19.46
11	21	**21**	**22**	21.77
12	23	24	**24**	24
13	25	**26**	**26**	26
14	27	**28**	**28**	28
15	29	**30**	**30**	30
16	31	**32**	**32**	32
17	33	**34**	**34**	34
18	35	**36**	**36**	36
19	37	**38**	**39**	38.8
20	39	**40**	**40**	40

Table 12: Testing Results in SE_d by Using Refinement

of nodes in the graphs in Table 13 is 200, and the number of nodes in the graphs in Table 14 is 500. When the number of edges increases to about two or three times the number of nodes in a graph, the new heuristic tends to generate the optimal broadcast time.

The graphs in tables 15 and 16 are generated by using the GT-ITM Transit-Stub model. The numbers of nodes in the graphs in Table 15 and 16 are 600 and 1056 respectively. In each graphs from Table 15, there are 3 transit domains, and each transit domain has 8 nodes. Each transit node is connected to three stub domains, while each stub domain has 8 nodes. The edge probabilities between the nodes in the transit domains and stub domains are both 0.5. The graphs in Table 16 have the same properties as the graphs in Table 15, except that there are 4 transit domains in the former graphs.

In tables 17 and 18, part of the testing results from the Tiers model are presented. For the graphs in Table 17, $N_w = 1$, $N_m = 10$, $N_l = 5$, $S_w = 5$, $S_m = 10$ and $S_l = 5$, which determine that the number of nodes in the graphs is 355. For the graphs in Table 18, $N_w = 1$, $N_m = 10$, $N_l = 10$, $S_w = 5$, $S_m = 10$ and $S_l = 10$, which determine that the number of nodes in the graphs is 1105.

Sometimes it is hard to determine the optimal broadcast time of these graphs generated by the models. Therefore, the testing results cannot be validated by comparing with the optimal broadcast time. The performance of the heuristic in these models will be validated by comparing the results with those from other heuristics in

Random		Origin			Refinement			
P	Edge	Min	Max	Ave	P	Min	Max	Ave
0.015	316	11	12	11.76	4	10	11	11.87
0.016	346	10	11	10.45	1	10	11	10.17
0.017	373	10	11	10.04	1	10	10	10
0.018	388	9	11	10.03	3	9	10	9.05
0.019	391	11	12	11.14	1	11	12	11.09
0.02	411	9	10	9.31	2	9	9	9
0.021	410	9	10	9.51	1	9	10	9.58
0.022	423	9	10	9.08	1	9	9	9
0.024	475	8	9	8.97	1	8	9	8.98
0.025	484	8	9	8.91	1	8	9	8.66
0.026	507	8	9	8.95	8	8	9	8.69
0.028	555	8	9	8.51	1	8	9	8.26
0.03	595	8	9	8.33	2	8	9	8.24

Table 13: GT-ITM Pure Random $N = 200$

Chapter 6.

Random		Origin			Refinement			
P	Edge	Min	Max	Ave	P	Min	Max	Ave
0.008	1003	11	12	11.04	2	10	11	10.78
0.009	1198	10	11	10.36	1	10	11	10.25
0.01	1238	10	11	10.13	1	10	10	10
0.011	1413	10	10	10	1	10	10	10
0.012	1481	10	10	10	1	10	10	10
0.014	1725	10	10	10	1	10	10	10
0.015	1830	10	10	10	6	9	10	9.99
0.016	2074	10	10	10	1	9	10	9.97

Table 14: GT-ITM Pure Random $N = 500$

TS		Origin			Refinement			
N	Edge	Min	Max	Ave	P	Min	Max	Ave
0	1206	14	15	14.57	2	15	15	15
1	1232	13	14	13.8	1	14	14	14
2	1222	14	16	15.58	1	15	15	15.77
3	1231	13	15	13.69	1	14	15	14.28
4	1190	14	14	14	1	14	14	14
5	1169	13	15	13.84	1	14	14	14
6	1280	13	14	13.38	1	13	14	13.34
7	1200	15	15	15	2	16	16	16
8	1247	14	16	14.65	1	14	14	14
9	1219	14	16	14.7	1	15	15	15

Table 15: GT-ITM TS $N = 600$

TS		Origin			Refinement			
N	Edge	Min	Max	Ave	P	Min	Max	Ave
0	2230	16	17	16.07	2	16	17	16.48
1	2084	17	17	17	1	17	18	17.5
2	2209	15	17	16.2	2	15	16	15.61
3	2219	16	17	16.39	7	16	18	16.41
4	2151	15	17	15.42	1	15	17	15.65
5	2230	15	17	15.63	2	16	16	16
6	2115	17	17	17	1	16	16	16
7	2187	15	16	15.64	3	15	16	15.15
8	2155	18	20	18.72	1	19	19	19
9	2220	15	17	15.48	1	15	16	15.67

Table 16: GT-ITM TS $N = 1056$

Tiers $N = 355$						Origin			Refinement			
Edge	R_w	R_m	R_l	R_{mw}	R_{lm}	Min	Max	Ave	P	Min	Max	Ave
354	1	1	1	1	1	17	17	17	2	17	17	17
414	1	1	1	2	2	14	14	14	2	14	14	14
474	1	1	1	3	3	13	15	13.97	2	13	14	13.08
357	2	1	1	1	1	17	17	17	2	17	17	17
477	2	1	1	3	3	14	15	14.58	2	14	15	14.54
535	2	1	1	4	4	15	15	15	2	15	15	15
422	3	2	1	2	2	14	16	14.98	2	14	16	15.36
482	3	2	1	3	3	13	15	14.13	4	14	15	14.31
541	3	2	1	4	4	14	15	14.69	5	14	15	14.21

Table 17: Tiers $N = 355$

Tiers $N = 1105$						Origin			Refinement			
Edge	R_w	R_m	R_l	R_{mw}	R_{lm}	Min	Max	Ave	P	Min	Max	Ave
1104	1	1	1	1	1	24	24	24	2	24	24	24
1214	1	1	1	2	2	21	23	22.01	2	21	22	21.84
1324	1	1	1	3	3	21	23	22.29	3	22	23	22.51
1447	1	1	1	4	4	21	22	21.23	3	21	21	21
1106	2	2	1	1	1	24	24	24	2	24	24	24
1216	2	2	1	2	2	21	23	22.03	2	21	22	21.84
1326	2	2	1	3	3	21	23	22.31	5	22	23	22.55
1449	2	2	1	4	4	20	22	20.67	3	20	21	20.69
1110	3	2	1	1	1	23	23	23	2	23	23	23
1220	3	2	1	2	2	21	22	21.31	2	21	22	21.23
1331	3	2	1	3	3	21	21	21	2	20	20	20
1458	3	2	1	4	4	21	23	21.73	2	21	22	21.03

Table 18: Tiers $N = 1105$

Chapter 6

Comparing with Three Previously-known Heuristics

6.1 The Round_Heuristic

In this section, the testing results of the new heuristic will be compared with the performance of three previously-known heuristics, which were introduced in Chapter 2.

6.1.1 In Commonly Used Topologies

The authors of [3] present the best results in the CCC_d graph, the $deBruijn$ graph, the $Shuffle\ Exchange$ graph and the $butterfly$ graph. The new heuristic also has been tested in those graphs. For purposes of comparison, the best results of the new

heuristic are listed along with those from [3] in Table 19. All the results for $d \leq 14$ are the same as those in [3] except for four cases: in two cases the new heuristic gives better results (mentioned by *) and in two cases the result from RBJS is better (mentioned by -). Three of the four differences happened in the *butterfly* graph, where $d = 10$, $d = 12$ and $d = 14$.

While the authors of [3] have values for $d \leq 14$, the new heuristic has been tested in graphs with d up to twenty in the *deBruijn* and *Shuffle Exchange* graph, and d up to sixteen in the *butterfly* and CCC_d graph. In fact, the new heuristic generates new upper bounds on broadcast time for BF_d, CCC_d when $15 \leq d \leq 16$, and for $UB(2,d)$ and SE_d when $15 \leq d \leq 20$. In addition, the new heuristic has been tested on *Hypercube* and *Star* graph, and it generates again new upper bounds.

6.1.2 In Three Graph Models

Among the approximately two hundred graphs generated by the three graph models, in only one graph, the new heuristic gave a broadcast time that was one more than that obtained by using the algorithm from [3]. In all other cases, the new heuristic obtained either the same broascast time as in [3] or better. In the *Pure Random* model the new heuristic gives better broadcast time in about twenty-five percent of the cases, while in the $Transit-Stub$ model the new heuristic gives better broadcast time in more than forty percent of the cases. The new heuristic worked better under the Tiers model, as it gave a smaller broadcast time in about sixty percent of the

k	CCC_d		SE_d		BF_d		UB_d	
	RH	A	RH	A	RH	A	RH	A
3	6	6	5	5	5	5	4	4
4	9	9	7	7	7	7	5	5
5	11	11	9	9	9	9	7	6*
6	13	13	11	11	10	10	8	8
7	16	16	13	13	12	12	9	9
8	18	18	15	15	14	14	11	11
9	21	21	17	17	16	16	12	12
10	23	23	19	19	17	18⁻	14	14
11	26	26	21	21	19	19	15	15
12	28	28	24	24	22	21*	17	17
13	31	31	26	26	23	23	18	18
14	33	33	28	28	24	25⁻	20	20
15		36		30		27		21
16		39		32		29		23
17				34				25
18				36				26
19				38				28
20				40				29

Table 19: The Minimum Testing Results of Round_Heuristic and the New Heuristic

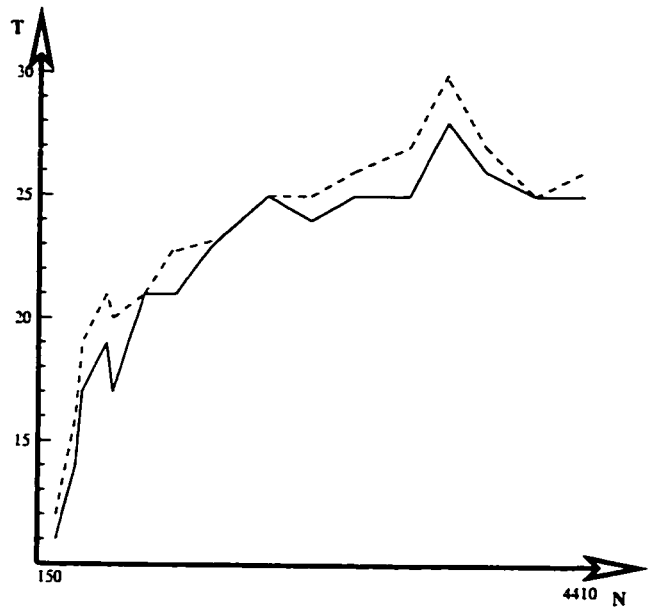
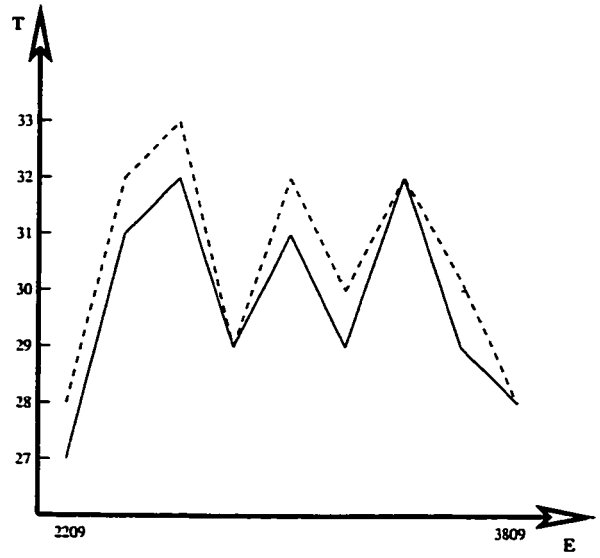


Figure 27: Tiers

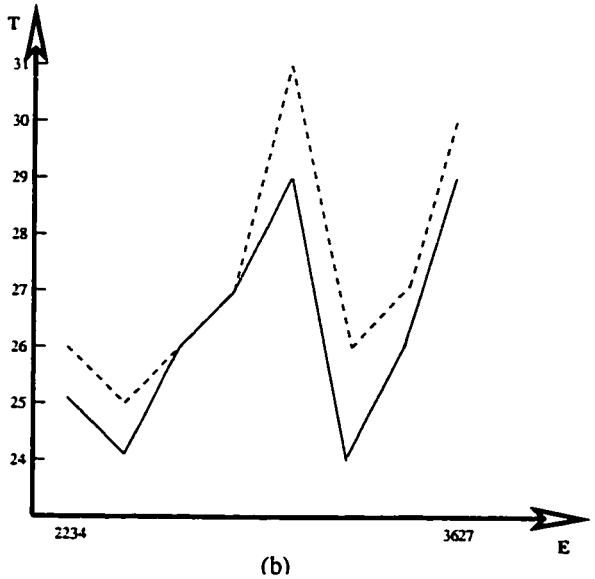
cases.

The figures in this section represent testing results of the new heuristic and the algorithm from [3] in the three models. In these figures, the solid line stands for the testing results of the new heuristic, and the dashed line represents the testing results of the algorithm from [3].

Figure 27, Figure 28 and Figure 29 present some of the testing results in the Tiers model. The horizontal scale in Figure 27 represents the number of nodes in the tested graphs. The vertical scale in Figure 27 refers to the broadcast time. It is possible that a bigger graph has a smaller broadcast time, because the number of edges and the topological structure of a graph also play important roles in determining the broadcast time.



(a)



(b)

Figure 28: Tiers $N = 2205$

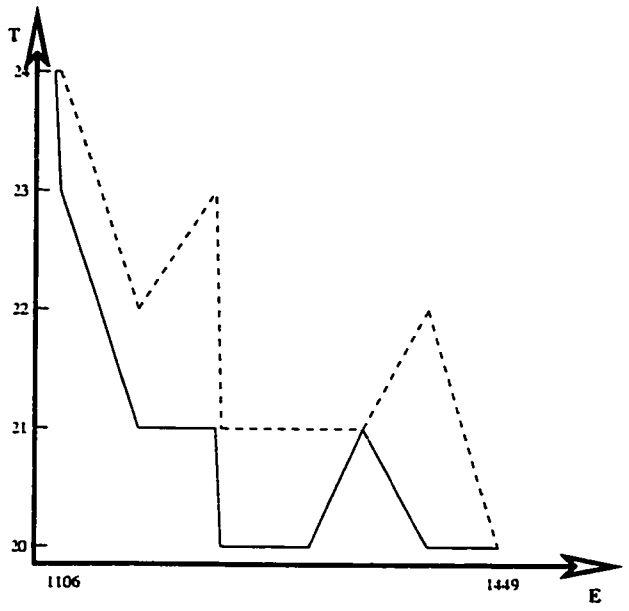


Figure 29: Tiers $N = 1105$

Figure 28 and Figure 29 list the testing results in graphs that have 2205 and 1105 nodes respectively. The horizontal scale in Figure 28 and Figure 29 stands for the number of edges in the tested graphs, while the vertical scale in them refers to the broadcast time.

In both Figure 28 (a) and (b), the tested graphs have 2205 nodes. The number of edges of graphs in Figure 28 (a) is increased by adding connections between a LAN and a MAN. The graphs in Figure 28 (b) are generated by adding several extra edges between a WAN and a MAN to graphs in Figure 28 (a). The WAN, MAN and LAN stand for the Wide Area Network, Metropolitan Area Network and Local Area Network respectively (see Chapter 2). Therefore, in Figure 28 (a) and (b), the broadcasting time of graphs with approximately identical number of edges could be

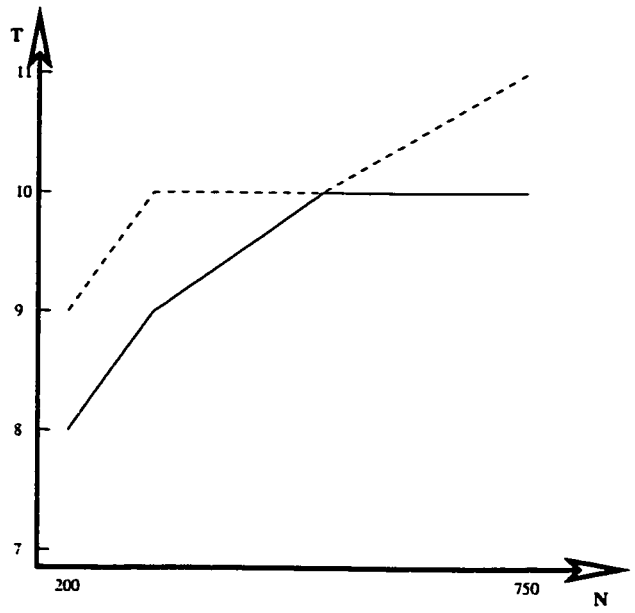


Figure 30: GT-ITM Pure Random

dramatically different.

The testing results in the GT-ITM Pure Random model are presented in Figure 30. The vertical scale in Figure 30 stands for the broadcast time of tested graphs. The horizontal scale in Figure 30 is the number of nodes. By using the Pure Random model, about ten different graphs for each value of the number of nodes were generated. Then one of the results is picked up randomly to put in the figure.

The testing results in the GT-ITM TS model is provided in Figure 31 and Figure 32. The vertical scales in these figures stand for the broadcast time of tested graphs. The horizontal scales refer to the number of edges in tested graphs.

In the point shown by the solid filled circle, the new heuristic performs one round

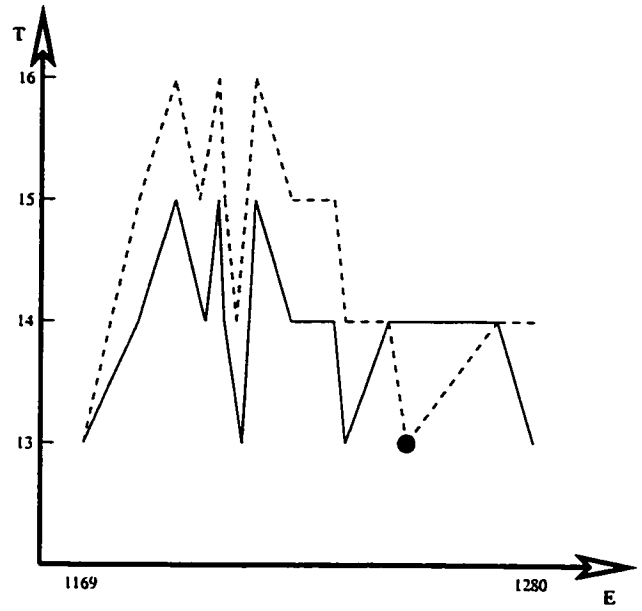


Figure 31: GT-ITM TS $N = 600$

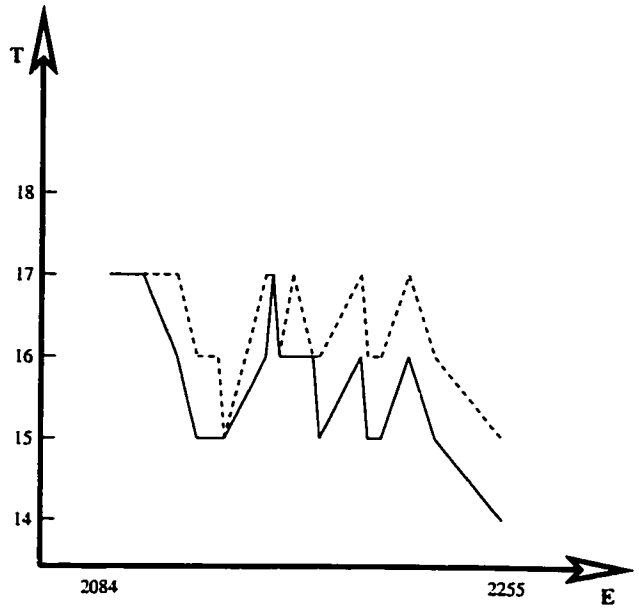


Figure 32: GT-ITM TS $N = 1056$

worse. This is the only instance where the new heuristic generates a worse result than the result from the algorithm in [3].

6.1.3 Summary

We can see that the new heuristic performs almost the same as the Round_Heuristic [3] in several commonly used topologies, but it performs better in these three graph design models. Moreover, the new heuristic is much faster. The time complexity of one round of the new heuristic is $O(m)$, while the time complexity of one round of the Round_Heuristic is $O(n^2 \cdot m)$. This is why the new heuristic could be tested in big graphs. In fact, the complexity of the Round_Heuristic does not include the time used to calculate the maximum weighted matching in a graph. A low time complexity could also improve the performance in practice. For instance, given a graph and a period of time, a fast heuristic would run more times, which allows the possibility of obtaining a better broadcast time.

In comparison with the Round_Heuristic, the new heuristic is very simple, and more importantly, it is easy to implement. As already introduced in the Chapter 3, the matching algorithms to perform the new heuristic are greedy algorithms. We do not need to calculate a maximum weighted matching in a graph. In the Round_Heuristic, finding such a matching is necessary, which makes the implement action more complicated.

As the authors mentioned, the quality of the Round_Heuristic heavily depends on

the choice of the parameters. There are two parameters in this heuristic. A range from 0.25 to 60 of one parameter is used to test the heuristic. In practice, it is hard to determine the best parameter, and this could degrade the performance of the heuristic. When using the new heuristic, there is no parameter at all. Even when using refinement, there is only one parameter. Moreover, as tested, the values of the parameter are the integer numbers between 1 and 8. Combined with the low time complexity of the new heuristic, it is easy to try these values in practice.

6.2 Two Other Heuristics

The new heuristic is compared with the MBT heuristic and the poise heuristic (see Chapter 2) in this section. The bound of broadcast time by using the MBT heuristic is $3 \cdot \sqrt{n} + Diam(v) + b(v)$, in which n is the number of nodes in the graph, $Diam(v)$ is the diameter of the originator and $b(v)$ is the real broadcast time. The bound of the poise heuristic is $O(b(G)\frac{\log n}{\log \log n})$. The maximum testing results of the new heuristic will be compared with the bound of MBT and poise heuristic in the commonly used topologies introduced in Chapter 2. It becomes impossible to compare the performances in the three graph models, because it is difficult to determine the diameters and real broadcast times of these graphs generated by these models. For the purpose of demonstration, only the situation that d, D or n is equal to 9 is considered. In case the optimal broadcast time is unknown, the lower bound is used as the optimal broadcast time. Table 20 presents the results. The maximum testing result of the

Graph	N	diameter	b(v)	MBT	poise	Max
H_9	512	9	9	87	26	10
$UB(2,9)$	512	9	12	90	34	13
BF_9	4608	13	15	230	51	17
SE_9	512	17	17	103	48	19
S_9	362880	12	19	1837	83	20
CCC_9	4608	20	21	245	71	22

Table 20: Bounds of Two Heuristics and Testing Results of the New Heurisitc

new heuristic is denoted by MAX in Table 20. We can see that the maximum testing results are much less than the MBT bound and the poise bound.

Chapter 7

Conclusion and Future Work

In this paper, a new heuristic is presented for broadcasting in arbitrary networks. In each round, the heuristic weighs all uninformed nodes in a graph. Then it produces a match between informed and uninformed nodes, after which, the message will be distributed between the mates.

This heuristic gives the optimum broadcast time in several simple topologies, such as the complete graph, the ring, the grid and the tree.

This heuristic performs well in practice. In several commonly used topologies, it works the same as the best known heuristic from [3] (Round_Heuristic). However, the new heuristic outperforms the Round_Heuristic in the graphs generated by three graph models. One of these models produces pure random graphs, and the other two models generate graphs that represent real networks. For only one graph out of the two hundred tested, the performance of the new heuristic is worse, while in forty

percent of the other cases, the new heuristic gives a better broadcast time.

The most important advantage of this heuristic is the low time complexity, which is $O(m)$ in each round. The time complexity of the heuristic in [3] is $O(n^2 \cdot m)$ in each round. The low time complexity makes it possible for the new heuristic to be tested in bigger graphs. As a result, the new upper bounds for several topologies were generated in practice.

It would be interesting to find theoretical bounds of this heuristic on arbitrary networks and on the commonly used topologies. Based on the experimental results, it seems that this heuristic should have good bounds on these tested graphs. However, the bounds still have not been found except for several simple topologies.

Bibliography

[1] S. Akers, D. Harel and B. Krishnamurthy, The star graph: an attractive alternative to the n-cube, *Proceedings of the International Conference on Parallel Processing*, PA, 1987, pp. 393-400.

[2] W. Aiello, F. Chung and L. Lu, Random evolution in massive graphs, *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001, pp. 510-519.

[3] R. Beier, J. F. Sibeyn, A Powerful Heuristic for Telephone Gossiping. *The 7th International Colloquium on Structural Information & Communication Complexity (SIROCCO 2000)*, L'Aquila, Italy, 2000, pp. 17-36.

[4] J.-C. Bermond and P. Fraigniaud, Broadcasting and gossiping in de Bruijn networks, *SIAM J. Comput. 23* 1994, pp. 212-225.

[5] J.-C. Bermond and C. Peyrat, Broadcasting in de Bruijn networks, *Proceeding 19th S-E Conference Combinatorics, Graph Theory, and Computing, FL Congressus Numerantium 66.* 1988, pp. 283-292.

[6] P. Berthomé, A. Ferreira and S. Perennes, *Tech. Rept.* LIP, ENS-Lyon, France, 1992.

[7] S. Djelloul, Etudes de certains réseaux d'interconnexion: structures et communications, PhD Thesis, Université Paris-Sud, Orsay, 1992.

[8] M. B. Doar, A Better Model for Generating Test Networks, *IEEE GLOBECOM'96*, London, UK, 1996.

[9] A. Farley and S. Hedetniemi, Broadcasting in grid graphs. *Proc. Eighteenth SE Conf. on Combinatorics, Graph Theory and Computing.* Utilitas Mathematica, Winnipeg, 1978, pp. 275-288.

[10] P. Fraigniaud and E. Lazard, Methods and problems of communication in usual networks. *Discrete Appl. Math.* 53, 1994, pp. 79-133.

[11] P. Fraigniaud and S. Vial, Approximation Algorithms for Broadcasting and Gossiping. *Journal of Parallel and Distributed Computing.* 43(1), 1997, pp. 47-55.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. F. Freeman, San Francisco, 1979.

[13] A. Harary and A. J. Schwenk, The communication problem on graphs and digraphs. *J. Franklin Inst.* 297, 1974, pp. 491-495.

[14] H. A.Harutyunyan and B. Shao, A Heuristic for Broadcasting. *IASTED International Conference on Communications and Computer Networks (CCN 2002)*, Cambridge, USA, 2002, pp. 360-365.

[15] S.M Hedetniemi, S.T. Hedetniemi and A.L. Liestman, A survey of gossiping and broadcasting in communication networks. *Networks* 18, 1996, pp. 319-349.

[16] J. Hromkovic, R. Klasting, B. Monien and R. Peine, Dissemination of information in interconnection networks (broadcasting and gossiping), *Tech. Rept.*, 71, Universitat Paderborn, 1993.

[17] J. Hromkovic, R. Klasing, B. Monien and R. Peine, Dissemination of Information in Interconnection Networks. *Combinatorial Network Theory.* D.-Z. Du, D.F. Hsu(eds.), Kluwer Academic Publishers, 1996, pp. 125-212.

[18] R. Klasing, B. Monien, R. Peine and E. A. Stöhr, Broadcasting in butterfly and deBruijn networks. *Discrete Appl. Math.* 53, 1994, pp. 183-197.

[19] G. Kortsarz and D. Peleg, Approximation Algorithms for Minimum Time Broadcast. *SIAM J. Discrete Math.* 8, 1995, pp. 401-427.

[20] T. Leighton, *Introcution to Parallel Algorithms and Architectures: Array-Trees-Hypercubes*, Morgan-Kaufmann Publishers, San Mateo, California, 1992.

[21] D.S. Meliksetian and C.Y.R. Chen, Communication aspects of the cube-connected cycles, *Proceedings of the International Conference on Parallel Processing.* 1990, pp. I579-I580.

[22] V.E. Mendia and D. Sarkar, Optimal broadcasting on the star graph, *IEEE Trans. Parallel Distrib. System.* 3, 1992, pp. 389-396.

[23] R. Ravi, Rapid Rumor Ramification: Approximating the minimum broadcast time. *35th Symposium on Foundation of Computer Science*, 1994, pp. 202-213.

[24] P. Scheuermann and G. Wu, Heuristic Algorithms for Broadcasting in Point-to-Point Computer Network. *IEEE Transactions on Computers*. C-33(9), 1984.

[25] P.J. Slater, E.J. Cockayne and S.T. Heditniemi, Information dissemination in trees. *SIAM J.Comput.* vol. 10, no. 4, 1981, pp. 692-701.

[26] E. W. Zegura, K. Calvert, and S. Bhattacharjee, How to Model an Internetwork. *IEEE INFOCOM*, San Francisco, CA, 1996.