2010 16th International Conference on Parallel and Distributed Systems

# Broadcasting Algorithm via Shortest Paths

Hovhannes A. Harutyunyan
Department of Computer Science and Software Engineering
Concordia University
Montreal, QC, Canada
e-mail: haruty@cse.concordia.ca

Wei Wang
Department of Computer Science and Software Engineering
Concordia University
Montreal, QC, Canada
e-mail: w_wei12@encs.concordia.ca

***Abstract*—In this paper, we present a new heuristic that generates broadcast schemes in arbitrary networks. The heuristic gives optimal broadcast time for ring, tree and grid if the originator is on the corner. Extensive simulations show that our new heuristic outperforms the best known broadcast algorithms for two different network models representing Internet and ATM networks. It also allows to generate broadcast time of networks of bigger size because its time complexity, O(|E|), is lower compared to the complexities of the other algorithms. The last advantage of the heuristic is that every node is informed via a shortest path from the originator.**

***Keywords-algorithm; broadcasting; network; heuristic; shortest path***

## I. INTRODUCTION

Broadcasting is the process of dissemination of information in an interconnection network, in which the information originated at one node is transmitted to all the other nodes in the network. The broadcasting process is composed of a series of calls. Each call takes one unit of time, and involves two vertices (the sender and the receiver). At each time unit every informed vertex can send the message to only one of its adjacent nodes. A time unit is also called a round, and the number of rounds is used to measure the broadcast time. Given a connected network of $n$ processors represented by a graph $G = (V, E)$, and an originator vertex $v$, the broadcast time of vertex $v$, denoted by $b(v)$, is the minimum broadcast time originated from $v$. The broadcast time of $G$, $b(G) = max\{b(v) \mid v \in V\}$.

To find the optimal broadcast time and broadcast scheme in an arbitrary graph is NP-complete [23]. Up to now, a great deal of effort has been dedicated on this problem, and some approximation algorithms and heuristics to find minimum broadcast times in this or slightly different models have been presented. [1,5,6,8,11,12,13,14,15,16,17,19,22,24]. Given a graph $G = (V, E)$ and the originator $u$, the heuristic in [19] returns a broadcast scheme whose performance is at most $b(u, G) + O(\sqrt{(|V|)})$ rounds. Theoretically, the best approximation is presented in [5]. Their approximation algorithm generates a broadcast protocol with $O(\log|V|/\log\log|V|)b(G)$ rounds. The two best existing heuristics are the Round Heuristic (RH) [1] and the Tree Based Algorithm (TBA) [9]. The complexity of RH is $O(R|V|^2|E|)$, where R is the number of broadcast rounds. TBA offers the best broadcast time in most interconnection topologies as well as some network models from ns-2 simulator, a widely used simulator for networking research. The run time of TBA is $O(R|E|)$.

In [10] we present a random heuristic called the New Tree Based Algorithm, whose time complexity is $O(|E|)$. In this paper, we will present a semi-random version of that heuristic for the message broadcasting in arbitrary networks, which also guarantees the message passing through the shortest path from the originator to every other vertex in the network. The time complexity of the new algorithm is also $O(|E|)$, same as the complexity of the random version and lower than that of all other existing heuristics. The final broadcast scheme is based on a spanning tree, thus we call this new heuristic the New Tree Based Algorithm (NTBA). NTBA performs well on several commonly used interconnection networks, such as Shuffle-Exchange graphs, and Cube-Connected Cycles. Besides the commonly used topologies, NTBA also works as well as RH from [1] and TBA from [9] in most network models from ns-2 simulator. Another advantage of NTBA is of course its low complexity. The last property should be mentioned is that NTBA informs each node via a shortest path from the originator, which is essential for some systems, e.g., hop-limited systems, load balancing systems, and some systems demanding a small total number of message duplications.

The remainder of this paper is structured as follows: the details of NTBA are formally described in Section 2, some theoretical results is presented in Section 3, and the test results on interconnection topologies and network models from ns-2 simulator are presented in Section 4.

## II. THE NEW TREE BASED ALGORITHM (NTBA)

First we present several definitions. Here, we denote the shortest distance from vertex u to vertex v by $D(u, v)$.

**Definition 1**. Layer: Given an originator $o$ and any vertex $v$, the layer of $v$, denoted by $L(v)$, is the shortest distance from o to $v$. Thus, $L(v) = D(o, v)$.

1521-9097/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPADS.2010.110

**Definition 2**. Layer graph: Given a graph $G = (V, E)$, $G_L = (V_L, E_L)$ is called the layer graph of $G$, where $V_L = V$ and for any edge $(u, v) \in E$, $(u, v) \in E_L$ *iff* $L(v) = L(u) + 1$.

**Definition 3**. Child and parent: If vertex $u$ and $v$ are neighbors and $L(v) = L(u) + 1$, then $v$ is a child of $u$, and $u$ is the parent of $v$.

**Definition 4**. Descendant: Any child of vertex $u$ is its descendant. Any of the children of the descendants of $u$ is also a descendant of $u$.

**Definition 5**. Estimated Broadcast Time: In order to estimate the broadcast time of any vertex $v$ in graph $G$, we employ the concept of estimated broadcast time, denoted by $EB(v)$, which can be calculated by the following recursion.

a) $EB(v)$ is equal to 0, if vertex $v$ has no children.
b) If vertex $v$ has $k$ children, $c_1, c_2, \ldots, c_k$, and all these children are in the order such that $EB(c_i) \geqslant EB(c_i+1)$, then $EB(v) = max\{EB(c_i) + i\}$, where $1 \leqslant i \leqslant k$.

Here, a linear algorithm will be introduced to calculate $EB(v)$ if $EB(c_i)$ is given, where $c_i$ is the child of vertex $v$, and $1 \leqslant i \leqslant k$. The complexity of this algorithm is $O(k)$.

Algorithm *Calculate_EB*:

1. Find $max\{EB(c_i)\}$ of $v$, and denote it by $MAX$.
2. Create $k$ buckets, and number them from 0 to $k$-1.
3. Any child $c$ of $v$, if $MAX - i \geqslant EB(c) > MAX - i - 1$, put $c$ into the $i$-th bucket. Here, $SUM(i)$ is employed to denote the number of elements in the first $i$ buckets, and $MIN(i)$ is to denote the minimal value in the $i$-th bucket.
4. Finally, $EB(u) = max\{SUM(i) + MIN(i)\}$, for $0 \leqslant i < k$.

The proof of the last step is as follows: Given a vertex $v$ with its $k$ children, $c_1, c_2, \ldots, c_k$, which are ordered such that $EB(c_i) \geqslant EB(c_i+1)$, then $EB(v) = max\{EB(c_i) + i\}$, for $1 \leqslant i \leqslant k$. $EB(c_i) + i$ is named order-weight of $c_i$. As $MAX - i \geqslant EB(c) > MAX - i - 1$ for any child $c$ in the $i$-th bucket, the maximum difference among all $EB$s in this bucket is less than 1. Therefore, in the $i$-th bucket, the child with the minimum $EB$ has the maximum order-weight, which is equal to $SUM(i) + MIN(i)$. Thus, $max\{SUM(i)+MIN(i)\}$ is the maximum order-weight of all the children, which is $EB(v)$.

**Lemma 1**. For any vertex $v$ in a tree, $EB(v)$ is exactly the time that vertex $v$ broadcasts the message to all of its descendants.

Proof : Vertex $v$ and its descendants make up of a tree that rooted at $v$. If vertex $v$ has $n$ children, $c_1, c_2, \ldots, c_n$, and without loss of generality $EB(c_1) \geqslant EB(c_2) \geqslant \ldots \geqslant EB(c_n)$. Thus the optimal broadcast scheme of vertex $v$ is that $v$ sends the message to child $c_i$ at round $i$. Since each vertex has one and only one parent in a tree, $EB(v)$ equals to $max\{EB(c_i) + i\}$ which gives only one possible value. Hence, it is easy to conclude that $EB(v)$ is the broadcast time of root $v$.

Fig. 1 illustrates these definitions. Vertex $a$ is the originator, thus $L(a) = 0$. Vertex $c$ is a child of $a$, since $L(c) = D(a, c) = 1$. Meanwhile, vertices $d$, $e$ and $f$ are adjacent to $c$ and $L(d) = L(e) = L(f) = 2$, thus vertex $c$ is also the parent of $d$, $e$ and $f$. By the definition, we can say that vertices $c$, $d$, $e$ and $f$ are all the descendants of $a$. Fig. 1(b) shows the layer graph $G_L$, in which vertex $a$ is on layer 0, vertices $c$ and $d$ are both on layer 1, while $d$, $e$, $f$ are all on layer 2.

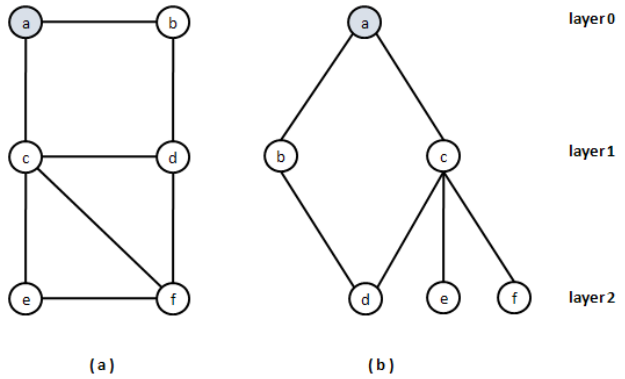


Figure 1. The example of the layer graph. (a) Original graph $G$ (b) Layer graph $G_L$.

Given a graph $G = (V, E)$ and originator $o$, NTBA intends to obtain a spanning tree of small broadcast time and every vertex connects to the root $o$ through the shortest path. Thus, how to generate such a spanning tree turns into the main problem. Here, we can take advantage of one primary property of the tree, that is every vertex in the tree can only have one parent. Therefore, if we can pick and determine only one parent for each child in the layer graph, we will achieve the spanning tree with the shortest paths from root to any other member. We call the action of selecting parent for children the matching. There are different ways to pick parent for each child according to the matching scheme. In [10], a pure random matching strategy between parents and children is introduced, where each child in the layer graph is just randomly matched with a parent. In the following section, a semi-random version of NTBA will be described in details, where the condition of each parent is taken into account. Here, the estimated broadcast time is used to weight a parent's condition. After matching all pairs of parent and child, a spanning tree for broadcasting is generated. The details of the algorithm are as the follows.

**ALGORITHM** *NTBA (the Semi-Random version)*:

1. Start from the originator $o \in G$, run breadth-first search (BFS) algorithm. Remove all edges that haven't been traversed during BFS, by which a layer graph $G_L$ is constructed. Here, we assume there are $k$ layers in $G_L$, and $n_l$ vertices at layer $l$, where $0 \leqslant l \leqslant k - 1$.
2. Label each vertex $v_i$ on layer $k - 1$ with weight $EB(v_i)$.
3. In layer graph $G_L$, for each layer $l$ starting from $k - 2$ to 1, call procedure *SRM*.
4. Finally, $EB(o)$ is the broadcast time of originator $o$ in graph $G$.

**PROCEDURE** *SRM*:
Input: All vertices on layer $l$ and layer $l + 1$.

1. On layer $l$, for each parent $p_i$ starting from $p_1$ to $p_{nl}$, we take following actions:
   (a) match $p_i$ with those children who have different weights, and remove all edges that connect these children with other parents;
   (b) label $p_i$ with $EB(p_i)$ not including those unmatched children.
2. For each unmatched child, perform operations below:
   (a) match it with the parent of the smallest weight, and remove its edges connecting to other parents;
   (b) update the weight *EB(p)* for its matched parent $p$.

According to definition 5, we can claim that children with the same estimated time may increase their parent's estimated time. *SRM* is a greedy algorithm, which tries to make each parent to adopt children of different weights. Fig. 2-6 illustrates an example how the NTBA algorithm works. Fig. 2 shows the layer graph $G_L$ after performing BFS in the original graph $G$. Vertex $a$ is the originator. All the edges connecting nodes on the same layer have been removed. Fig. 5 shows the spanning tree generated by the algorithm, which has the minimum broadcast time 5.

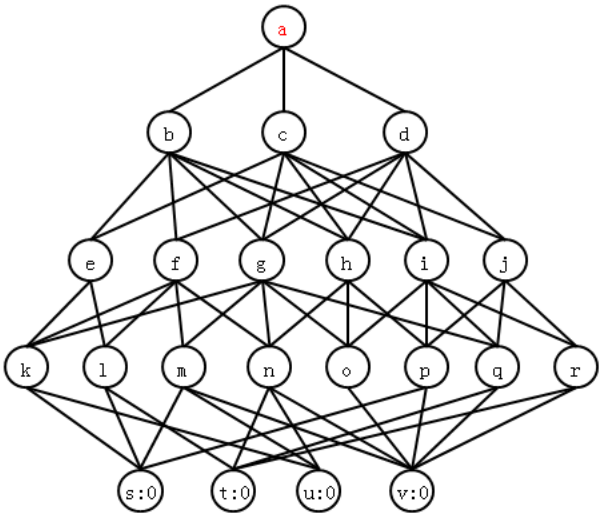


Figure 2. Layer graph $G_L$.

At beginning, all the vertices on the layer 4 are label with weight 0 (see Fig. 2). Then perform procedure *SRM* between layer 3 and layer 4. Vertex $k$ has two children, vertex $s$ and vertex $u$. Since vertices $s$ and $u$ have the same weight 0, $k$ picks $s$ as its child, and puts $u$ aside to be determined later. Similar, vertex $l$ takes vertex $t$, vertex $m$ takes vertex $u$, and vertex $n$ takes vertex $v$. Every child on the layer 4 has been matched with a parent, and removes all edges connected with other unmatched parents. Label all the vertices on the layer 3 with their weight *EB(v)*. Fig. 3 shows the results after performing *SRM* between layer 3 and layer 4. All vertices on the layer 3 have been label with a weight.

Then we go to the next two layers, layer 2 and layer 3. Parent $e$ picks child $k$, because $k$ and $l$ have the identical weight. In a similar way, vertex $f$ picks vertex $m$, vertex $g$ picks vertex $n$ and vertex $o$, vertex $h$ picks vertex $p$, vertex $i$ picks vertex $q$, and vertex $j$ picks vertex $r$. However, there is still a child $l$ left, who has not been taken by any parent. We calculate the weight of every vertex on the layer 2 based on those matched edges. Fig. 4 shows such situation. Dashed line *(e, l)* and *(f, l)* are to be decided, thus they are not taken into account when calculating the weight of vertex $e$ and vertex $f$. Now vertex $l$ knows that both of its parents have the same weight, it just randomly picks vertex $f$ as its parent and removes the edge connected with vertex $e$. Then update the weight of vertex $f$ (see Fig. 5).

Again, the procedure *SRM* is performed in a similar way between layer 1 and layer 2. Parent $b$ takes children $e$, $f$ and $h$, parent $c$ takes children $g$ and $i$, and parent $d$ takes child $j$. Fig. 6 shows the final spanning tree generated by perform the NTBA. The weight of the originator $a$ ($EB(a)$) is equal to 5, which is also the broadcast time of vertex $a$.

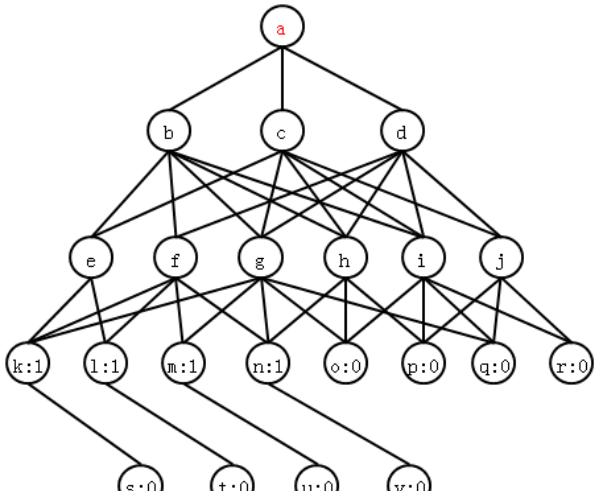


Figure 3. Procedure *SRM* performed between layer 3 and layer 4.

Even though having more operations than the random version in [10], the Semi-Random version of NTBA doesn't increase the time complexity at all. As usual, breadth-first search and generation of the layer graph terminate in $O(|E|)$. The procedure *SRM* doesn't exceed that bound either. In the first step of *SRM*, each child has to be determined if it has the same weight as others. When using binary tree to store the weight and binary search to check the weight, it's complexity is $O(log1) + O(log2) + \ldots + O(log|V|) = O(log|V|)$. Besides, the calculation of $EB(pi)$ takes $O(deg(pi))$ for each parent $p_i$, and totally it is of

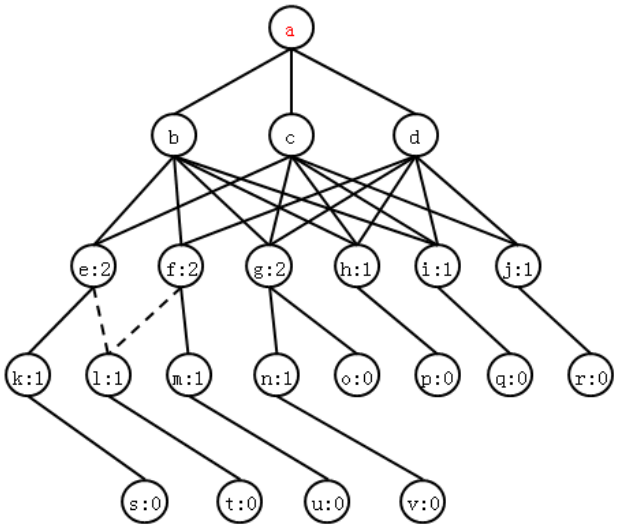


Figure 4. Procedure *SRM* performed between layer 2 and layer 3.

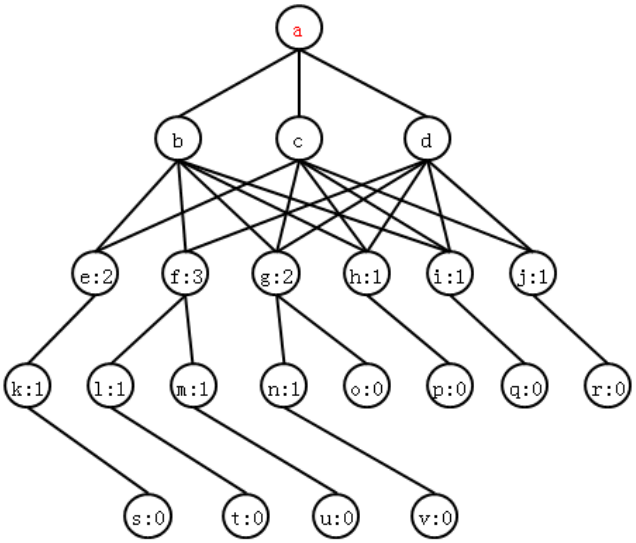


Figure 5. Procedure *SRM* performed between layer 3 and layer 2.

complexity O(|*E*|) again. Hence, the complexity of the first step of *SRM* is O(log|*V*| + |*E*|) = O(|*E*|). In the second step of *SRM*, searching a parent of the smallest weight for each child $c_i$ runs in O($deg(c_i)$), thus totally its complexity is O(|*E*|). Then, updating *EB*(v) for any single parent can be done in const time, and the complexity of updating all parents is O(|*V*|). We can conclude that the procedure SRM is of complexity O(|*E*|), and the total complexity of Semi-Random algorithm is also O(|*E*|).

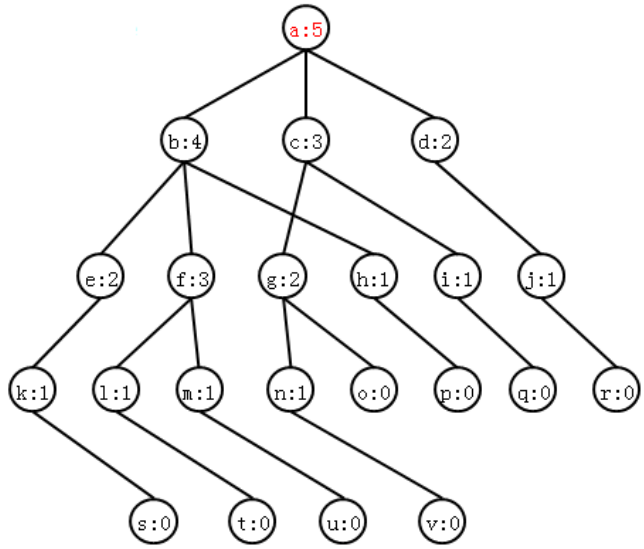


Figure 6. The spanning tree for broadcasting.

## III. THEORETICAL RESULTS

It is obvious that applying different matching strategies in a layer graph can lead to different broadcast schemes and broadcast time. In [19], a pseudo-polynomial algorithm is introduced to solve the Minimum Weight Cover (MWC) problem, which is called the MWC algorithm. This algorithm can be performed between each pair of adjacent layers in the layer graph to generate a spanning tree for broadcasting. In particular, if applying the MWC algorithm in a layer graph with 3 layers, the broadcast time generated by this algorithm is 2-approximation.

First we will describe the Minimum Weight Cover problem.

Let $G(V_1, V_2, A, w)$ be a bipartite graph with bipartition $(V_1, V_2)$, edge set $A$, and a weight function $w: A \rightarrow Z^+$ on the edges, and no isolated vertices. Each vertex $v_1 \in V_1$ is called a parent, and each vertex $v_2 \in V_2$ is called a child. If a control function $F: V_2 \rightarrow V_1$, where $F(v_2) = v_1$ implies $(v_1, v_2) \in A$, and we say that $v_1$ controls $v_2$. For every server $v \in V_1$, the children dominated by $v$ is denoted by $D_1(v) \ldots D_k(v)$, and the edges connecting $v$ with its children is denoted by $e^v_i = (v, D_i(v))$. Without loss of generality, all the children dominated by $v$ are in the order such that $w(e^v_i) \geqslant w(e^v_{i+1})$ for $1 \leqslant i \leqslant k$.

**Definition.** The MWC problem. Given a bipartite graph $G(V_1, V_2, A, w)$, determine a control function $F: V_2 \rightarrow V_1$ whose weight $W(F)=\max_{v \in V_1}\{\max_i\{i + w(e^v_i)\}\}$ is minimal. The function $F$ is called the minimum control function for $G$.

**Theorem 1.** After performing the MWC algorithm between each pair of adjacent layers in any layer graph with 3 layers, the broadcast time generated by the MWC is at most twice the minimum broadcast time.

**Proof.** There is always one node, the originator, on the layer 0, and we assume that there are $n$ nodes on the layer 1. Denote these vertices by $v_1, v_2,\ldots,v_n$. Based on the spanning tree generated by performing the MWC algorithm, without loss of generality, all the $n$ nodes are sorted in the descending order of their weights, i.e., $WA_1 \geqslant WA_2 \geqslant \ldots \geqslant WA_n$, where $WA_i$ is the weight of vertex $v_i$ on the layer 1, for all $1 \leqslant i \leqslant n$. Let $b(A,G)$ be the broadcast time of the spanning tree generated by the MWC algorithm in the layer graph G. According to Definition 5 and Lemma 1, $b(A, G) = max\{i + WA_i\}$, $1 \leqslant i \leqslant n$, and we can conclude that $b(A, G) \leqslant WA_1 + n$.

Now consider an optimal broadcast scheme which generates a minimum broadcast time of the layer graph $G$. Based on the optimal broadcast scheme, all $n$ nodes on the layer 1 are assumed to be in the order such that $WO_1 \geqslant WO_2 \geqslant \ldots \geqslant WO_n$, where $WO_i$ is the weight of $v_i$ on the layer 1 in optimal situation, $1 \leqslant i \leqslant n$. We denote the optimal broadcast time of the spanning tree of the layer graph $G$ by $b(O, G)$. Since $b(O, G) = max\{i + WO_i\}$, $1 \leqslant i \leqslant n$, we have $b(O, G) \geqslant max(WO_1 + 1, WO_n + n)$. From the MWC algorithm it follows $WA_1$ is the minimum possible value for $v_1$, thus $WO_1 \geqslant WA_1$. As $WO_1 \geqslant WA_1$ and $WO_n \geqslant 0$, we can conclude that $b(O, G) \geqslant max\{WA_1+1, n\}$.

Now we have that $b(A, G) \leqslant WA_1 + n$ and $b(O, G) \geqslant max\{WA_1+1, n\}$.

When $WA_1+1 \geqslant n$, then clearly $b(O, G) \geqslant WA_1+1$, thus $b(A, G) / b(O, G) \leqslant (WA_1 + n)/ (WA_1+1) \leqslant (2n-1)/ n \leqslant 2$.

When $WA_1+1 \leqslant n$, then it is easy to see that $b(O, G) \geqslant n$, thus $b(A, G)/b(O, G) \leqslant (WA_1 + n) / n \leqslant (n - 1 + n)/n \leqslant 2n-1/n \leqslant 2$.

We can see that in any case $b(A, G)/b(O, G)$ is less than or equal to 2, thus the broadcast time generated by the MWC algorithm in any layer graph with 3 layers is at most twice the minimum broadcast time. Thus, for a graph with diameter 6 MWC gives 2–approximation for minimum broadcast problem.

## IV. SIMULATION RESULTS

In this section, we present our experimental results in several interconnection topologies such as HyperCube, Cube-Connected cycle, Butterfly graph, deBruijn graph, and Shuffle-Exchange graph, as well as some network models from ns-2 simulator.

### TEST RESULTS IN COMMONLY USED TOPOLOGIES

Tables 1-5 show the broadcast time obtained by our algorithm NTBA, as well as algorithms RH, TBA and C from [1], [9] and [21] respectively in the following interconnection networks: HyperCube ($H_d$), Butterfly graph ($BF_d$), Shuffle-Exchange graph ($SE_d$ ), deBruijn($DB_d$) and Cube-Connected Cycle ($CCC_d$). *Low* and *Up* stands for the best known theoretical lower and upper bounds, respectively. *Opt* is the optimal broadcast time of a graph. All these bounds and optimal broadcast times are from [2,3,7,18,20]. In all tables the minimum broadcast time among all the algorithms RH, TBA, C, NTBA is indicated in bold. Newly generated best results by NTBA are starred.

The simulation result shows that in general our algorithm NTBA performs well in Cube-Connected Cycles and Shuffle-Exchange graphs. Besides, NTBA also has a not bad performance in Butterfly graph and deBruijn graph. However, our algorithm does not fit for those graphs where most vertices have high degrees, such as Hypercubes. Since NTBA has very low time complexity, it can be used for broadcasting in larger graphs. Here, we present some new values of broadcast time in interconnected networks for bigger values of dimension. Another advantage of our algorithm is that each member in the network receives the message via the shortest path from the originator.

TABLE I. TEST RESULTS IN $H_D$

| d | Opt | TBA | C | NTBA |
|---|---|---|---|---|
| 5 | 5 | **5** | **5** | 5 |
| 6 | 6 | **6** | 7 | 7 |
| 7 | 7 | **7** | 8 | 9 |
| 8 | 8 | 9 | 9 | 11 |
| 9 | 9 | 10 | 10 | 14 |
| 10 | 10 | 11 | 12 | 15 |
| 11 | 11 | 12 | - | 18 |
| 12 | 12 | 13 | - | 20 |
| 13 | 13 | 14 | - | 22 |
| 14 | 14 | 15 | - | 25 |
| 15 | 15 | 16 | - | 27 |
| 16 | 16 | 17 | - | 30 |
| 17 | 17 | 18 | - | 32 |
| 18 | 18 | 19 | - | 34 |
| 19 | 19 | 20 | - | 37 |
| 20 | 20 | 21 | - | 39 |

TABLE II. TEST RESULTS IN $CCC_D$

| d | Low | Up | RH | TBA | NTBA |
|---|---|---|---|---|---|
| 3 | 6 | 7 | **6** | **6** | **6** |
| 4 | 9 | 9 | **9** | **9** | **9** |
| 5 | 11 | 12 | **11** | **11** | **11** |
| 6 | 13 | 14 | **13** | **13** | 14 |
| 7 | 16 | 17 | **16** | **16** | **16** |
| 8 | 18 | 19 | **18** | **18** | 19 |
| 9 | 21 | 22 | **21** | **21** | **21** |
| 10 | 23 | 24 | **23** | **23** | 24 |
| 11 | 26 | 27 | **26** | **26** | 27 |
| 12 | 28 | 29 | **28** | **28** | 29 |
| 13 | 31 | 32 | **31** | **31** | 32 |
| 14 | 33 | 34 | **33** | **33** | 34 |
| 15 | 36 | 37 | | **36** | 37 |
| 16 | 38 | 39 | | **39** | 40 |
| 17 | - | - | | - | 43* |

| | | | | | |
|---|---|---|---|---|---|
| 18 | - | - | | - | 46* |

TABLE III. TEST RESULTS IN $SE_D$

| d | Opt | RH | TBA | C | NTBA |
|---|---|---|---|---|---|
| 3 | 5 | **5** | **5** | **5** | **5** |
| 4 | 7 | **7** | **7** | **7** | **7** |
| 5 | 9 | 9 | 9 | **8** | 9 |
| 6 | 11 | 11 | 11 | **10** | 11 |
| 7 | 13 | 13 | 13 | **12** | 13 |
| 8 | 15 | 15 | 15 | **14** | 15 |
| 9 | 17 | 17 | 17 | **16** | 18 |
| 10 | 19 | 19 | 19 | **18** | 20 |
| 11 | 21 | **21** | **21** | - | 22 |
| 12 | 23 | **24** | **24** | - | **24** |
| 13 | 25 | **26** | **26** | - | **26** |
| 14 | 27 | **28** | **28** | - | **28** |
| 15 | 29 | | **30** | - | **30** |
| 16 | 31 | | **32** | - | **32** |
| 17 | 33 | | **34** | - | **34** |
| 18 | 35 | | **36** | - | **36** |
| 19 | 37 | | **38** | - | **38** |
| 20 | 39 | | **40** | - | **40** |
| 21 | - | | - | - | 42* |

TABLE IV. TEST RESULT IN $DB_D$

| d | Low | Up | RH | TBA | NTBA |
|---|---|---|---|---|---|
| 3 | 4 | 6 | **4** | **4** | **4** |
| 4 | 6 | 8 | **5** | **5** | **5** |
| 5 | 7 | 9 | 7 | **6** | 7 |
| 6 | 8 | 11 | **8** | **8** | 8 |
| 7 | 10 | 12 | **9** | **9** | 10 |
| 8 | 11 | 14 | **11** | **11** | 12 |
| 9 | 12 | 15 | **12** | **12** | 13 |
| 10 | 14 | 17 | **14** | **14** | 15 |
| 11 | 15 | 18 | **15** | **15** | 17 |
| 12 | 16 | 20 | **17** | **17** | 19 |
| 13 | 18 | 21 | **18** | **18** | 20 |
| 14 | 19 | 23 | **20** | **20** | 22 |
| 15 | 20 | 24 | | **21** | 24 |
| 16 | 22 | 26 | | **23** | 26 |
| 17 | 23 | 27 | | **25** | 28 |
| 18 | 24 | 29 | | **26** | 30 |
| 19 | 26 | 30 | | **28** | 32 |
| 20 | 27 | 32 | | **29** | 33 |

TABLE V. TEST RESULTS IN $BF_D$

| d | Low | Up | RH | TBA | NTBA |
|---|---|---|---|---|---|
| 3 | 5 | 5 | **5** | **5** | **5** |
| 4 | 7 | 7 | **7** | **7** | 8 |
| 5 | 8 | 9 | **9** | **9** | 10 |
| 6 | 10 | 11 | **10** | **10** | 12 |
| 7 | 11 | 13 | **12** | **12** | 14 |
| 8 | 13 | 15 | **14** | **14** | 16 |
| 9 | 15 | 17 | **16** | **16** | 18 |
| 10 | 16 | 19 | **17** | 18 | 20 |
| 11 | 18 | 21 | **19** | **19** | 22 |
| 12 | 19 | 23 | 22 | **21** | 24 |
| 13 | 21 | 25 | **23** | **23** | 26 |
| 14 | 23 | 27 | **24** | 25 | 28 |
| 15 | 24 | 29 | | **27** | 30 |
| 16 | 26 | 31 | | **29** | 32 |
| 17 | - | - | | - | 34* |
| 18 | - | - | | - | 36* |

### TEST RESULTS IN THREE GRAPH MODELS

In order to obtain reliable intervals of the simulation results, a statistical concept called confidence interval is employed in this section, which is used to indicate the reliability of an estimate. A confidence interval is always qualified by a particular confidence level, usually expressed as a percentage; thus one speaks of a ”95% confidence interval”. For each graph during the simulation, we run the NTBA 100 times. As a result, 100 samples for each graph are collected, then 99% confidence interval of the population is computed based on the sample mean and sample standard deviation. In the following tables, *99%CI* denotes the 99% confidence interval. If the *99%CI* field is blank, that means all the simulation results in that graph are identical, or the results in that graph do not have an interval.

The test results from ns-2 simulator are listed in the Tables 6-8. Here, three different network models are considered: GT-ITM Pure Random [25], GT-ITM Transit-Stub (TS) [25] and Tiers [4].

The Tiers 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. 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 vertices in a domain share routing information. Just like the real Internet, interconnected domains compose the graphs generated by GT-ITM Transit-Stub. GT-ITM Pure Random is a standard random graph model. Considering each pair of vertices, 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 to real networks.

Tables 6-8 present the simulation results for GT-ITM Pure Random model, TS model as well as Tiers models. The numerical results of these tables show that the performance of NTBA works well in most ns-2 network models except for GT-ITM Pure Random model. In TS model, all of the three algorithms have similar performances. However, Table 8 shows that NTBA performs especially better in Tiers model than the other two. Also, NTBA works much faster than other existing algorithms.

TABLE VI. TEST RESULTS IN GT-ITM PURE RANDOM MODEL

| Vertices | Edges | RH | TBA | NTBA | 99%CI |
|---|---|---|---|---|---|
| 500 | 1725 | **10** | **10** | 13 | 13.76-13.95 |
| 750 | 2099 | **11** | **11** | 13 | 14.24-14.69 |

TABLE VII. TEST RESULTS IN GT-ITM TS MODEL

| Vertices | Edges | RH | TBA | NTBA | 99%CI |
|---|---|---|---|---|---|
| 600 | 1169 | 14 | **13** | **13** | 13.2-13.45 |
| 600 | 1190 | 14 | 14 | **13*** | 14.27-14.78 |
| 600 | 1219 | 15 | 14 | **13*** | 13.21-13.46 |
| 600 | 1231 | 14 | **13** | **13** | 13.51-13.76 |
| 600 | 1247 | **13** | 14 | 14 | - |
| 600 | 1280 | 14 | **13** | 14 | - |
| 1056 | 2115 | 17 | **16** | **16** | 16.25-16.5 |
| 1056 | 2142 | 16 | **15** | **15** | 15.42-15.67 |
| 1056 | 2177 | 18 | 17 | **16*** | 16.23-16.48 |
| 1056 | 2185 | 16 | 16 | **15** | - |
| 1056 | 2219 | 17 | 16 | **15*** | 15.33-15.58 |
| 1056 | 2230 | 16 | **15** | **15** | 15.4-15.65 |

TABLE VIII. TEST RESULTS IN TIERS MODEL

| Vertices | Edges | RH | TBA | NTBA | 99%CI |
|---|---|---|---|---|---|
| 1105 | 1110 | 24 | 23 | **21*** | - |
| 1105 | 1214 | 22 | **21** | **21** | - |
| 1105 | 1331 | **20** | **20** | **20** | - |
| 1105 | 1447 | 22 | **21** | 22 | - |
| 2210 | 2234 | 26 | **25** | **25** | - |
| 2210 | 2833 | 27 | 27 | **24*** | |
| 2210 | 3009 | 32 | 31 | **24*** | 24.23-24.47 |
| 2210 | 3609 | 30 | 29 | **23*** | 23.67-23.94 |

## V. CONCLUSION AND FUTURE WORK

We have introduced the new algorithm NTBA and its experimental results in both commonly used topologies and network models from ns-2 simulator. NTBA provides good performance in general, and particularly works well in large graphs representing Internet type networks (GT-ITM Transit-Stub) and ATM type networks (Tiers). Furthermore, NTBA informs every node of the network via a shortest path from the originator. The last advantage of NTBA is its low complexity-$O(|E|)$, while other algorithms have $O(|V||E|)$ and $O(|V|^3|E|)$, complexities in the worst case. All the above good properties make NTBA a very good heuristic to be used in practice. In the future work, one could improve the matching strategy between children and parents, design an approximation algorithm for the message broadcasting in the layer graph, and obtain a theoretical bound on the broadcast time of the layer graph as well.

## REFERENCES


[1] R. Beier, J.F. Sibeyn, A powerful heuristic for telephone gossiping, the Seventh International Colloquium on Structural Information & Communication Complexity (SIROCCO 2000), L'Aquila, Italy, 2000, pp. 17-36.

[2] P. Berthomé, A. Ferreira, S. Perennes, Optimal information dissemination in star, pancake networks, IEEE Trans. Parallel Distributed Systems 7 (12) (1996) 1292–1300.

[3] S. Djelloul, Etudes de Certains Réseaux d'Interconnexion: Structures et Communications, Ph.D. Thesis, Université Paris-Sud, Orsay, 1992.

[4] M.B. Doar, A better model for generating test networks, in: IEEE GLOBECOM'96, London, UK, 1996.

[5] M. Elkin, G. Kortsarz, Sublogarithmic approximation for telephone multicast: path out of jungle, in: Symposium on Discrete Algorithms, Baltimore, MA, 2003, pp. 76–85.

[6] U. Feige, D. Peleg, P. Raghavan, E. Upfal, Randomize broadcast in networks. SIGAL International Symposium on Algorithms, 1990, pp. 128–137.

[7] P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks, Discrete Appl. Math. 53 (1994) 79–133.

[8] P. Fraigniaud, S. Vial, Approximation algorithms for broadcasting and gossiping, J. Parallel Distributed Comput. 43 (1) (1997) 47–55.

[9] H. A. Harutyunyan, B. Shao, An efficient heuristic for broadcasting in networks, J. Parallel Distrib. Comput. 66 (2006) 68-76.

[10] H. A. Harutyunyan, W. Wang, A random heuristic for message broadcasting in arbitrary networks, the 11th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT-2010).

[11] H. A. Harutyunyan. Minimum Multiple Message Broadcast Graphs, Networks, 47 (4) (2006) 218-224.

[12] H. A. Harutyunyan, A. L. Liestman, B. Shao, A Linear Algorithm for Finding the k-Broadcast Center, Networks, 53(3) (2009) 287-292.

[13] Hovhannes A. Harutyunyan, Arthur L. Liestman: k-Broadcasting in trees. Networks 38(3): (2001) 163-168.

[14] H. A. Harutyunyan, B. Shao, A Heuristic for k-Broadcasting in Arbitrary Networks. Seventh International Conference on Information Visualization, (IV 2003), 2003, pp. 287-293.

[15] H. S. Haroutunian, Minimal broadcast networks, Fourth International Colloquium on Coding Theory, Dilijan, Armenia, 1991, pp. 36-40.

[16] L H Khachatrian, H S Haroutunian, On optimal broadcast graphs, Fourth International Colloquium on Coding Theory, Dilijan Armenia, 1991, pp. 65-72.

[17] L. H. Khachatrian and H. S. Haroutunian, Minimal broadcast trees, XIV All Union School of Computing Networks, Minsk, 1989, pp. 36-40 (in Russian).

[18] R. Klasing, B. Monien, R. Peine, E.A. Stohr, Broadcasting in butterfly and deBruijn networks, Discrete Appl. Math. 53 (1994) 183–197.

[19] G. Kortsarz, D. Peleg, Approximation algorithms for minimum time broadcast, SIAM J. Discrete Math. 8 (1995) 401–427.

[20] A.L. Liestman, J.G. Peters, Broadcast networks of bounded degree, SIAM J. Discret Math. 1 (1988) 531–540.

[21] H. A. Harutyunyan, C. D. Morosan, On Two Properties of the Minimum Broadcast Time Function. International Conference on Information Visualization, (IV 2005), 2005, pp. 523-527.

[22] R. Ravi, Rapid rumor ramification: approximating the minimum broadcast time, in: 35th Symposium on Foundation of Computer Science, 1994, pp. 202–213.

[23] P. J. Slater, E.J. Cockayne, S.T. Heditniemi, Information dissemination in trees, SIAM J. Comput. 10 (4) (1981) 692–701.

[24] P. Scheuerman, G. Wu, Heuristic Algorithms for Broadcasting in Point-to-Point Computer Network, IEEE Trans. Comput. C-33 (9) (1984) 804–811.

[25] E.W. Zegura, K. Calvert, S. Bhattacharjee, How to model aninternetwork, in: IEEE INFOCOM, 1996.