2014 IEEE 28th International Conference on Advanced Information Networking and Applications

New Heuristic for Message Broadcasting in Networks

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

Cosmin Jimborean
Department of Computer Science and Software Engineering
Concordia University
Montreal, QC, Canada

***Abstract*—In this paper, we present a new heuristic that generates broadcast schemes in arbitrary networks. The heuristic gives optimal broadcast time for HyperCube, and best results for Cube-Connected Cycles and large Shuffle-Exchange graphs. Extensive simulations show that our new heuristic outperforms the best known broadcast algorithms for two different network models representing Internet generated using BRITE (Boston university Representative Internet Topology gEnerator). It also has a low time complexity, *O(|E|log|V|)*, which is lower compared to the complexities of most of the other good algorithms. The last advantage of the heuristic is that approximately one half of the nodes are informed via a shortest path from the originator, while the rest of the vertices receive the message via a path at most three hops longer.**

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

I. INTRODUCTION

In computer science, information dissemination encapsulates a set of problems related to the distribution of information within an interconnection network. One of the most researched computer science topics in the last years, efficient information dissemination is important for an increasing number of topics, such as parallel and distributed computing, internet networks, virtual social networks and telecommunication networks.

In the past, computer processing power was consistently increased by boosting processor clock speed from kilohertz to gigahertz. Recently, the increases of clock speed seemed to have reached their limit and with the cost of hardware decreasing dramatically, the trend to increase processing power is to build massive multi-processor systems. The size of multi-processor systems has exploded in the last few years and hundred-thousand processor systems have already been built. Under these conditions, the problems of information dissemination are especially important when designing such massive interconnection networks for parallel and distributed computing. In parallel and distributed computing, the ability of the processors in the interconnection network to communicate efficiently is crucial. To study these problems, an interconnection network is modeled as a connected undirected graph where processors are represented by the nodes of the graph and the communication links are represented by the edges of the graph [30].

Information dissemination includes problems related to broadcasting, accumulation and gossiping. The distribution of information from one to all is known as broadcasting. Gathering the information from all to one is known as accumulation. Gathering the information from all and distributing it to all is known as gossiping. In this paper we will focus on broadcasting.

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. As mentioned above, an interconnection network is modeled as a connected undirected graph $G = (V,E)$, where V is the set of vertices and E is the set of edges in the graph G. Using this model, the following abstract definition of broadcasting is given in [31].

"Let $G = (V, E)$ be a graph and let $v \in V$ be a node of G. Let v know a piece of information $I(v)$ which is unknown to all nodes in $V \setminus \{v\}$. The problem is to find a communication strategy (algorithm) such that all nodes in G learn this piece of information $I(v)$."[31]

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. It is obvious that for a given graph and a given originator, multiple broadcasting communication strategies exist. The efficiency of a communication strategy is measured by the number of communication rounds needed to distribute the information from the source vertex to all vertices. 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

1550-445X/14 $31.00 © 2014 IEEE
DOI 10.1109/AINA.2014.63

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 $O(R|E|)$. The heuristics presented in [10] and [26], called the New Tree Based Algorithm (NTBA) are also very good heuristics, whose time complexity is $O(|E|)$.

In this paper, we will present a new heuristic for the message broadcasting in arbitrary networks, which builds upon the idea of NTBA but applies a new non-random strategy to generate a spanning tree for broadcasting. The time complexity of the new heuristic is $O(|E|\log|V|)$, slightly higher than that of the NTBA, but still lower than that of all other existing heuristics. The new heuristic performs very well on several commonly used interconnection networks, such as Hypercube (optimal results), Shuffle-Exchange graphs, and Cube-Connected Cycles. In addition, in BRITE (Boston university Representative Internet Topology gEnerator) top-down hierarchical models the results are the best, much better than the NTBA from [26]. Besides the above, the new heuristic also works as well as RH from [1], TBA from [9] and NTBA from [26] in most network models from ns-2 simulator. Another advantage of the new heuristic is of course its low complexity. The last property that should be mentioned is that approximately one half of the vertices are informed via a shortest path from the originator, while the rest of the vertices receive the message via a path at most three hops longer, 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 the new heuristic are formally described in Section 2 and the test results on interconnection topologies and network models from ns-2 simulator are presented in Section 3.

II. THE NEW HEURISTIC

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, $L(v) = D(o, v)$.

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 EBs 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.

Figure 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.

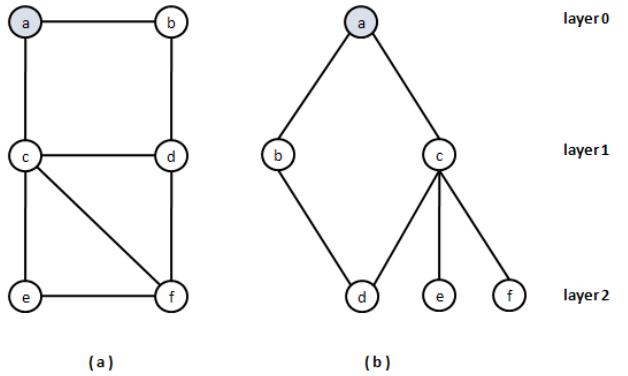

Figure 1 The example of the layer graph. (a) Original graph G (b) Layer graph G_L.

Given a graph $G = (V, E)$ and originator o, the new heuristic intends to obtain, in the first place, 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] and [26], two pure random and semi-random matching strategies between parents and children are introduced, where each child in the layer graph is just randomly or semi-randomly matched with a parent. In the following section, the new heuristic will be described in details, where the children selected for each parent are taken into account. Here, the estimated broadcast time is used to weight a parent's condition and the intuition is to match children with parents such that the estimated broadcast times of the parents are arranged in a series that is decreasing by exactly one at a time. After matching all pairs of parent and child, a spanning tree for broadcasting is generated. In the last step, the broadcasting scheme is improved by performing a broadcast and considering idle vertices at any round and potential edges that were removed during the construction of the layer graph. The details of the algorithm are as the follows.

ALGORITHM:

1. Construct the layer graph G_L of the arbitrary graph G by performing a breadth-first search starting from the originator and marking as inactive all the edges that have not been traversed during the breadth-first search.
2. Assuming that G_L has k layers, numbered from 0 to k -1, label all the vertices v_i of layer k -1 with $EB(v_i)$.
3. For each layer l starting from $k-2$ to 1, call procedure *Matching* to match children to one parent.
4. Perform procedure *Broadcast and Optimize* on the resulting spanning tree.
5. The broadcast time of o in graph G is the number of rounds returned by procedure *Broadcast and Optimize*.

PROCEDURE *Matching:*

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

1. Order the parents by decreasing number of children and denote the parents by $p_1, p_2, \ldots, p_n$ where p_1 has the most number of children and p_n the least number of children.
2. Start with an empty matching.
3. Add p_1 and all its children to the matching.
4. Match p_1 with all its children.
5. For each parent p_k, starting from p_2 to p_n, take the following actions:
   a) Add p_k and all its children to the matching.
   b) Match p_k with all its unmatched children at this step (those who have p_k as single parent)
   c) Reposition p_k in the matching such that the parents in the matching are ordered by decreasing estimated broadcast time $EB(p_i)$.
   d) Find the parent p_i which gives $b_{max} = max_{1 \le i \le k}\{i + EB(p_i)\}$ and denote this parent as p_{max}.
   e) If p_{max} is p_k then remove all unused edges and go to step 5.
   f) If the number of children common to both p_{max} and p_k is 0, then remove all unused edges and go to step 5.
   g) If $b_{max} = b(p_k)$ then remove all unused edges and go to step 5.
   h) From the children common to both p_{max} and p_k, choose the maximum weight child which has the same weight as another child of p_{max}. Remove the edge matching it to p_{max} and match it to p_k.
   i) Go to step 5.c).

PROCEDURE *Broadcast and Optimize:*

Input: A spanning tree, plus inactive edges between siblings (vertices on the same layer).

Output: A broadcast scheme in the spanning tree and the number of rounds required to broadcast a message in the spanning tree.

1. Inform originator o.
2. While there remain uninformed vertices do
   a) For each informed vertex v do
      i. If v has uninformed children, inform its next uninformed child.
      ii. Else if v has uninformed siblings, inform the next uninformed sibling that has maximum EB.
3. Output the broadcast scheme and the number of times the step 2 was performed.

According to definition 5, we can claim that children with the same estimated time may increase their parent's estimated time. Procedure Matching is a greedy algorithm, which tries to make each parent to adopt children of different weights. Its main goal is to assign children to parents in such a way that $max\ \{EB(p_i)\}$ becomes as small as possible. To achieve this, at each step, the algorithm tries to remove children from the parent with maximum EB at that point and assign them to the parent processed at that step. For a better understanding of the algorithm, an example of the procedure Matching is presented below. Figure 2 shows a bipartite graph between two layers of the layer graph G_L representing the input to procedure Matching. The estimated broadcast time for each child is labeled on the graph.

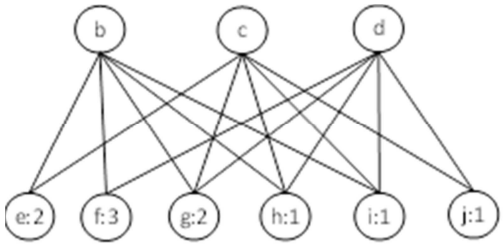

Figure 2 The original bipartite graph between two layers of the layer graph G_L

After ordering the parents by decreasing number of children, parent b with the most number of children is added to the matching and its estimated broadcast time is calculated. Figure 3 shows the matching at this point.

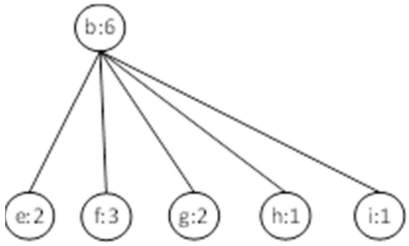

Figure 3 Parent b is added to the matching

Next step is shown in Figure 4. Parent d, the next one in order of decreasing number of children, is added to the matching together with all its children that have not already been added before. In this case child j is the only child of d which has not already been added, so j gets added to the matching and matched with d. The dashed lines represent unused edges that d adds to the matching and which could potentially be used in the next steps to minimize the estimated broadcast time of b.

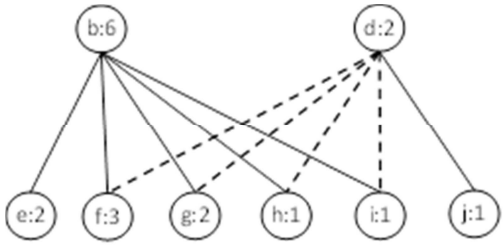

Figure 4 Parent d is added to the matching

At this point, d will try to minimize the estimated broadcast time of b by taking over one of b's children. Since the children with the same broadcast time are the ones that increase the estimated broadcast time of their parent, the algorithm chooses from the children with the same broadcast times, a child which has the max broadcast time among the children that are common to b and d. In Figure 5 we see that g is moved from b to d, hence the broadcast time of b decreases to 5 and the broadcast time of d increases to 3.

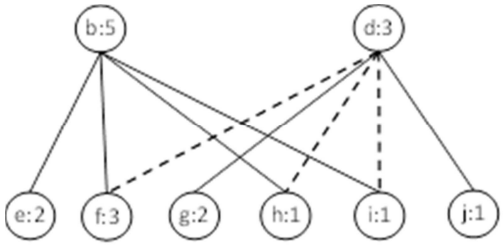

Figure 5 Child g is moved from parent b to d

Since the broadcast time of d is still less than the broadcast time of b, the previous step is repeated and child i gets moved from b to d, making the broadcast times of b and d equal to 4.

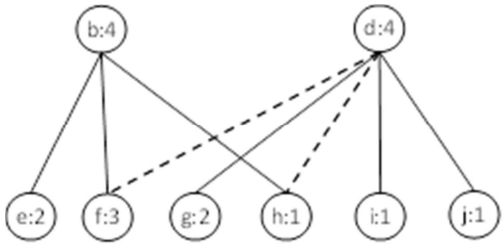

Figure 6 Child i is moved from parent b to parent d

Because the broadcast time of the current parent d is now equal to the broadcast time of b, the algorithm considers the processing of parent d complete and moves to the next parent. Parent c and all its children and edges are added to the matching. Since c does not have any children of its own, its initial broadcast time is 0.

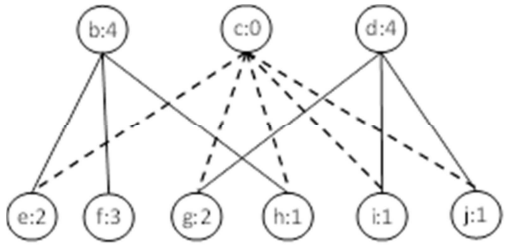

Figure 7 Parent c is added to the matching

At this point, in decreasing order of their estimated broadcast time $EB(p_i)$, parents b, d, c respectively have:

- $b(b) = i_b + EB(b) = 1 + 4 = 5$
- $b(d) = i_d + EB(d) = 2 + 4 = 6$
- $b(c) = i_c + EB(c) = 3 + 0 = 3$

hence $b_{max} = \max_{1 \le i \le k}\{i + EB(p_i)\}$ is given by parent d, with $b(d) = 6$, therefore parent c takes over child j from parent d and the resulting matching is shown in Figure 8.

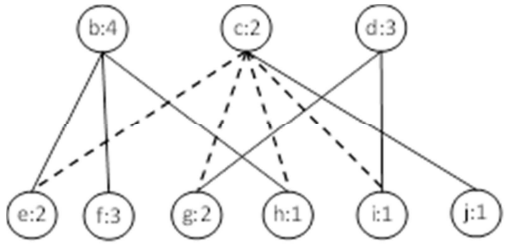

Figure 8 Child j is moved from parent d to parent c

Once again, in decreasing order of their estimated broadcast time $EB(p_i)$, parents b, d, c respectively have:

- $b(b) = i_b + EB(b) = 1 + 4 = 5$
- $b(d) = i_d + EB(d) = 2 + 3 = 5$
- $b(c) = i_c + EB(c) = 3 + 2 = 5$

Since c has now become one of the parents whose $b(c) = b_{max} = max_{1 \le i \le k}\{i + EB(p_i)\} = 5$, the processing of parent c is considered complete and since there are no more parents, the procedure terminates. The final matching is shown in Figure 9.

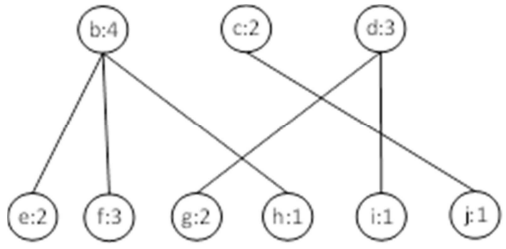

Figure 9 The final matching

The first step of the algorithm is the breadth-first search to construct the layer graph and its time complexity is $O(|V| + |E|) = O(|E|)$.

During the Procedure *Matching*, first, the parents must be sorted by decreasing number of children. Assuming there are k layers and n_i parents on each layer, where $1 \le i \le k$, the complexity of sorting the parents on one layer is $n_i \log n_i$ and the total complexity for all k layers is $\sum_{i=1}^{k} n_i \log n_i$. Since for all $i \in [1, k]$, $n_i \le |V|$, then of course $\sum_{i=1}^{k} n_i \log n_i \le \sum_{i=1}^{k} n_i \log|V| = |V| \log|V|$, hence in the worst case this step has $|V| \log|V|$ complexity.

In the second step, the matching requires calculating $EB(p_k)$ for each parent p_k, then finding the parent p_i which gives $max_{1 \le i \le k}\{i + EB(p_i)\}$ and moving children from one parent to another. Calculating $EB(v)$ has complexity $O(deg(v))$, where $deg(v)$ is the degree of vertex v, therefore the total complexity for all parents is $\sum_{v \in V} deg(v) = O(|E|)$. Finding the parent p_i which gives $max_{1 \le i \le k}\{i + EB(p_i)\}$ has complexity $O(n)$ where n is the number of parents on one layer, hence the total complexity for all layers is the number of vertices in the graph, $O(|V|)$. Moving children from parent p_{max} to parent p_k looks in the worst case at each one of the children of p_k and it searches for a child of p_{max} with the same weight as the respective child of p_k. Each binary search takes at most $O(log|V|)$, and needs to be done for at most all edges, hence the total complexity of this step is $O(|E| \log|V|)$.

The total complexity of the Procedure *Matching* is $O(|V| \log|V| + |E| \log|V|) = O(|E| \log|V|)$.

The last step is the Procedure *Broadcast and Optimize* which performs a broadcast using the spanning tree generated in the previous step and verifies if idle edges between siblings can be used to improve the broadcast time. In the worst case, the procedure goes through all the edges of the graph, hence the complexity of this procedure is $O(|E|)$.

Finally, the total complexity of the algorithm is the sum of the complexities of the procedures above $O(|E|) + O(|E| \log|V|) = O(|E| \log|V|)$.

III. 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 New Heuristic (NewH), as well as algorithms RH, TBA, C and NTBA from [1], [9], [21] and [26] 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 and NewH is indicated in bold.

The simulation result shows that in general our algorithm gives optimal results in HyperCube. It also performs very well in Cube-Connected Cycles and Shuffle-Exchange graphs. Besides, the new heuristic also has a not bad performance in Butterfly graph and deBruijn graph.

TABLE I. TEST RESULTS IN H_D

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

TABLE II. TEST RESULTS IN CCC_D

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

TABLE III. TEST RESULTS IN SE_D

d	Opt	RH	TBA	C	NTBA	NewH
3	5	**5**	**5**	**5**	**5**	**5**
4	7	**7**	**7**	**7**	**7**	**7**
5	9	9	9	**8**	9	9
6	11	11	11	**10**	11	11
7	13	13	13	**12**	13	13
8	15	15	15	**14**	15	15
9	17	17	17	**16**	18	18
10	19	19	19	**18**	20	20
11	21	**21**	**21**	-	22	22
12	23	**24**	**24**	-	**24**	**24**
13	25	**26**	**26**	-	**26**	**26**
14	27	**28**	**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**	-	**38**	**38**
20	39	**-**	**40**	-	**40**	**40**

TABLE IV. TEST RESULTS IN DB_D

d	Low	Up	RH	TBA	NTBA	NewH
3	4	6	**4**	**4**	**4**	**4**
4	6	8	**5**	**5**	**5**	**5**
5	7	9	7	**6**	7	7
6	8	11	**8**	**8**	8	**8**
7	10	12	**9**	**9**	10	10
8	11	14	**11**	**11**	12	12
9	12	15	**12**	**12**	13	13
10	14	17	**14**	**14**	15	15
11	15	18	**15**	**15**	17	17
12	16	20	**17**	**17**	19	19
13	18	21	**18**	**18**	20	20
14	19	23	**20**	**20**	22	22
15	20	24	**-**	**21**	24	24
16	22	26	**-**	**23**	26	26
17	23	27	**-**	**25**	28	28
18	24	29	**-**	**26**	30	30
19	26	30	**-**	**28**	32	32
20	27	32	**-**	**29**	33	34

TABLE V. TEST RESULTS IN BF_D

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

TEST RESULTS IN FOUR GRAPH MODELS

In order to obtain reliable intervals of the simulation results, a statistical concept called confidence interval is employed, 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 ran the simulation 20 times. Since all the simulation results we collected were identical, there was no need to compute the confidence intervals, which is one of the advantages of the new heuristic over the NTBA which has a random component and its results are reliable within a confidence interval.

The test results from ns-2 simulator are listed in the Tables 6-10. Here, 4 different network models are considered: GT-ITM Pure Random [25], GT-ITM Transit-Stub (TS) [25], Tiers [4] and BRITE Top-Down hierarchical models [27].

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. BRITE uses a top-down approach to generate hierarchical topologies. BRITE first generates an AS-level model using one of the available flat AS-level models, such as Waxman from [28] or Barabasi-Albert from [29]. Next, for each node in the AS-level model BRITE will generate a router-level model using a different generation model from the available flat models that can be used at the router-level. The router-level models are interconnected using one of four edge connection mechanisms, borrowed from the popular GT-ITM topology generator. The main goal is to gradually increase the set of edge connection methods with models that reflect what actually happens in Internet models.

Tables 6-10 present the simulation results for GT-ITM Pure Random model, TS model, Tiers models as well as BRITE models. The numerical results of these tables show that the new heuristic performs much better than NTBA in BRITE models. It also outperforms NTBA in the GT-ITM Pure Random model, where the best results are provided by TBA, but the time complexity of TBA is slightly worse than the new heuristic. The new heuristic works also well in the other ns-2 network models, showing results similar to existing heuristics. Also, the new heuristic works much faster than other existing algorithms.

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

Vertices	Edges	RH	TBA	NTBA	NewH
500	1413	11	**10**	13	11
500	1481	10	**10**	13	11
500	1725	**10**	**10**	13	11
500	1830	10	**9**	14	11
500	2074	**9**	**9**	15	10

TABLE VII. TEST RESULTS IN GT-ITM TS MODEL

Vertices	Edges	RH	TBA	NTBA	NewH
600	1169	14	**13**	**13**	14
600	1190	14	14	**13**	**13**
600	1219	15	14	**13**	15
600	1222	15	**14**	**14**	**14**
600	1231	14	**13**	**13**	14
600	1247	**13**	14	14	14
600	1280	14	**13**	14	15
1056	2115	17	**16**	**16**	**16**
1056	2142	16	**15**	**15**	16
1056	2169	17	17	**15**	**15**
1056	2177	18	17	**16**	**16**
1056	2185	16	16	**15**	16
1056	2219	17	16	**15**	**15**
1056	2230	16	**15**	**15**	16

TABLE VIII. TEST RESULTS IN TIERS MODEL

Vertices	Edges	RH	TBA	NTBA	NewH
1105	1110	24	23	**21**	23
1105	1214	22	**21**	**21**	23
1105	1216	22	21	21	**20**
1105	1220	22	**21**	**21**	**21**
1105	1331	**20**	**20**	**20**	21
1105	1447	22	**21**	22	**21**
1105	1449	21	**20**	22	21

TABLE IX. TEST RESULTS IN BRITE TOP-DOWN WAXMAN MODEL

Vertices	Edges	NTBA	NewH
400	1680	13	**12**
400	2092	13	**12**
400	2440	14	**12**
400	2671	16	**12**
400	2733	15	**11**
400	2755	18	**12**
1000	4080	16	**15**
1000	5100	**16**	**16**
1000	6108	16	**14**
1000	7116	17	**14**
1000	8117	18	**15**
1000	9122	19	**14**

TABLE X. TEST RESULTS IN BRITE TOP-DOWN BARABASI-ALBERT MODEL

Vertices	Edges	NTBA	NewH
400	1470	13	**12**
400	1785	13	**12**
400	2079	13	**12**
400	2352	14	**12**
400	2604	14	**12**

400	2835	15	**11**
1000	1977	22	**20**
1000	2934	21	**17**
1000	3870	18	**16**
1000	4785	17	**15**
1000	5679	17	**15**
1000	6552	17	**14**
1000	7404	17	**13**
1000	8235	18	**14**

IV. CONCLUSION AND FUTURE WORK

We have introduced a new heuristic for broadcasting in arbitrary networks and its experimental results in both commonly used topologies and network models from ns-2 simulator. The new heuristic provides good performance in general, and particularly works very well in HyperCube, Cube-Connected Cycles, Shuffle-Exchange graphs, and large graphs representing Internet type networks (BRITE). Furthermore, the new heuristic informs approximately half of the nodes of the network via a shortest path from the originator; while the rest of the vertices receive the message via a path at most three hops longer. The last advantage of the new heuristic is its low complexity-$O(|E|log|V|)$, while other algorithms have $O(R|E|)$ and $O(R|V|^2|E|)$, complexities in the worst case. All the above good properties make the new heuristic 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 an internetwork, in: IEEE INFOCOM, 1996.

[26] H.A. Harutyunyan and W. Wang. Broadcasting Algorithm Via Shortest Paths. in Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on. 2010.

[27] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: an approach to universal topology generation. in Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on. 2001.

[28] B.M. Waxman, Routing of multipoint connections. Selected Areas in Communications, IEEE Journal on, 1988. 6(9): p. 1617-1622.

[29] A.-L. Barabási and R. Albert, Emergence of Scaling in Random Networks. Science, 1999. 286(5439): p. 509-512.

[30] J. Hromkovič, R. Klasing, B. Monien, and R. Peine, Dissemination of information in interconnection networks (broadcasting & gossiping), in Combinatorial network theory. 1996, Springer. p. 125-212.

[31] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance (Texts in Theoretical Computer Science. An EATCS Series). 2005.