Available online at www.sciencedirect.com

SCIENCE DIRECT®

J. Parallel Distrib. Comput. 66 (2006) 68–76

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

An efficient heuristic for broadcasting in networks

Hovhannes A. Harutyunyan*,[1], Bin Shao

Department of Computer Science, Concordia University, Montreal, Quebec, Canada H3G 1M8

Received 20 June 2003; received in revised form 20 June 2005; accepted 22 June 2005
Available online 3 August 2005

Abstract

In this paper, we present a heuristic for broadcasting in arbitrary networks. This heuristic generates optimal broadcast time for ring, tree and grid graphs when the originator is a corner vertex. In practice, the new heuristic outperforms the best known broadcast algorithms for three different network models. The time complexity of one round of the heuristic is $O(|E|)$, where $|E|$ stand for the number of edges of the network.

Keywords: Algorithm; Broadcast; Network; Heuristic

1. Introduction

The performance of information dissemination often determines the overall efficiency of networks. One of the fundamental information dissemination problems is broadcasting, which is a process where a single message is sent from one vertex in a network to all other vertices. Normally, a network can be modeled as a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. At the beginning of broadcasting, only one vertex $u \in V$, called the *originator*, is informed. During each unit of time, each informed vertex passes the message to only one of its uninformed adjacent vertices. Such an action is referred as a *call*. Many calls can be performed in parallel, but a vertex can only call an adjacent vertex. A vertex can participate in only one call per unit of time. Broadcasting can be modeled as a sequence of parallel calls. A *round* is the set of parallel calls in a same time unit. We use the number of rounds to measure the broadcast time. The broadcast time $b(u, G)$, or simply $b(u)$, is the minimum broadcast time of graph G originated by vertex u. The broadcast time of graph G is defined as follows: $b(G) = max\{b(u, G) \mid u \in V\}$.

The problem of finding the optimal broadcasting schedule in an arbitrary graph is NP-hard [19]. Therefore, many papers have presented approximation algorithms to find minimum broadcast times [2,6,8,10,14,17,18]. Some of these papers give theoretical bounds on the broadcast time. Given $G = (V, E)$ and the originator u, the heuristic in [14] returns broadcast protocol whose performance is at most $b(u, G) + O(\sqrt{|V|})$ rounds. Theoretically, the best upper bound is presented in [6]. Their approximation algorithm generates a broadcast protocol with $O(\frac{\log(|V|)}{\log\log(|V|)})b(G)$ rounds. In this paper, we are interested in heuristics that perform well in practice. A heuristic for gossiping is presented in [2]. Gossiping is a more general problem. In gossiping, each vertex has a message that it needs to send to all other vertices. Every gossip scheme also gives a broadcast scheme for each vertex in a graph. The heuristic for broadcasting is only a by-product of [2]. However the heuristic in [2] is the best existing heuristic for broadcasting in practice.

This paper presents a new heuristic for broadcasting, which we call the tree based algorithm (TBA). This heuristic generates optimal broadcast time in rings, trees and in grid graphs when the originator is a corner vertex. In torus graph G, it gives an upper bound $b(G) + 3$. TBA generates the same broadcast time as the heuristic from [2] (with few exceptions) on several commonly used interconnection

* Corresponding author.
E-mail address: haruty@cs.concordia.ca (H.A. Harutyunyan).
[1] Author was supported by NSERC.

0743-7315/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2005.06.010

networks, such as the *de Bruijn*, the *Shuffle Exchange*, the *Butterfly* graphs and the *Cube-Connected Cycle*. However, TBA outperforms the heuristic in [2] on three graph models from a network simulator ns-2. Also, the time complexity of one round of TBA is $O(|E|)$, while the complexity of one round of the algorithm from [2] is $O(|V|^2 \cdot |E|)$.

The remainder of this paper is structured as follows: TBA is formally presented in Section 2, and the theoretical and experimental results concerning the heuristic are introduced in Sections 3 and 4, respectively.

2. The tree based algorithm (TBA)

In this section we will describe our algorithm. The algorithm always generates the optimal broadcast time of any vertex in an arbitrary tree. Thus, we called it TBA.

2.1. TBA and its complexity

In order to formally present TBA, we first give several definitions.

Definition 1. Bright border: The bright border $bb(t)$ is composed of those informed vertices that have uninformed neighbors at the beginning of round t.

Let $D(v, t)$ stand for the shortest distance from an uninformed vertex v to $bb(t)$ at round t.

Definition 2. Child and parent: Given an uninformed vertex u and its uninformed neighbor v, if $D(u, t) = D(v, t) + 1$, we say u is a child of v, and v is the parent of u.

Definition 3. Descendant: a child of vertex u is its descendant. Any of the children of the descendants of u is also a descendant of u.

Fig. 1 illustrates these definitions. In this example, vertex a is the originator. After three rounds, vertices in the shadowed area are still uninformed. The informed vertices with shadowed backgrounds belong to $bb(4)$. The distance between $bb(4)$ and the uninformed vertices with black backgrounds is one; and the distance between $bb(4)$ and the uninformed vertices n, o and p is two. Also, the distance between $bb(4)$ and the uninformed vertices q is three. So, vertices o and p are children of vertex j, and vertex q is a child of vertices o and p. We can also say that vertices o, p and q are descendants of vertex j.

The basic idea of TBA is to find a matching between the set of informed and uninformed vertices in each round, and then distribute the message between them. To achieve this, in each round, we first perform a modified BFS (breadth first search) from $bb(t)$ towards uninformed vertices. During this process, we label any uninformed vertex v with $D(v, t)$.

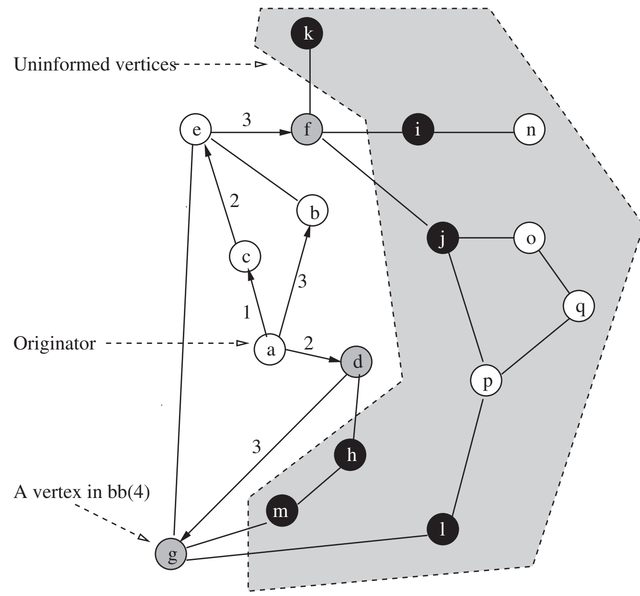

Fig. 1. Definitions in TBA.

Thus, the parent–child relationship among the uninformed vertices can be defined by these distances.

The optimal broadcasting in trees is discussed in [19]. Given an informed vertex v and its k uninformed neighbors $v_1, v_2, \ldots, v_k$, let $T(v_i, v)$ $(1 \leqslant i \leqslant k)$ stand for the tree that is rooted at v_i and does not contain v. Assume the k uninformed children are labeled such that $b(v_1, T(v_1, v)) \geqslant b(v_2, T(v_2, v)) \geqslant \cdots \geqslant b(v_k, T(v_k, v))$, then the optimal sequence of broadcast calls is that v first calls v_1, then v_2, then v_3, etc. By this broadcast scheme, after v is informed, all the vertices in any $T(v_i, v)$ $(1 \leqslant i \leqslant k)$ can be informed in $max\{b(v_i, T(v_i, v)) + i,\ 1 \leqslant i \leqslant k\}$ rounds.

The weight of a vertex in TBA is based on the strategy of the optimal broadcasting in trees. Let $w(u, t)$ stand for the weight of vertex u in round t. If u has no children, then $w(u, t) = 0$. If u has k children $v_1, v_2, \ldots, v_k$, and $w(v_1, t) \geqslant w(v_2, t) \geqslant \cdots \geqslant w(v_k, t)$, then $w(u, t) = max\{w(v_i, t) + i, 1 \leqslant i \leqslant k\}$. After that, we find a matching between the set of informed and uninformed vertices. We use a heuristic with time complexity $O(|E|)$ to find the matching. The heuristic tries to bring the number of matched pairs of vertices to a maximum; given this, it tries to maximize the weights of matched vertices. Finally, every matched informed vertex sends the message to its mate.

In TBA, procedures *Calculate_weight* and *Calculate_match* determine weights of all uninformed vertices and the matching in each round, respectively. Given the weights of all k children of a vertex v, procedure *Weight* returns the weight of v in time $O(k)$. This procedure is intended for both integers and decimals. The procedure *Weight* is based on bucket sorting. In case all the weights are integers, the procedure could be simpler. However, in the refinement of TBA, the weights could be fractional numbers. The procedure *Calculate_weight* starts with assigning weights to the vertices that have no children. Then it assigns weights to all uninformed vertices recursively by calling the procedure *Weight*. Procedure *Weight* takes $O(d)$ time to calculate the weight of a vertex with degree d. Thus, the time needed to calculate the weights of all the vertices is $\sum_{i=1}^{n} O(d_i)$. Since $\sum_{i=1}^{n} d_i = 2|E|$, the time complexity of the procedure *Calculate_weight* is $O(|E|)$. The procedure *Calculate_match* approximately computes a maximum weighted matching. All the vertices in $bb(t)$ are saved in a group of linked lists. The operations used in the procedure *Calculate_match* are similar to *build*(), *deletemin*() and *decreasekey*() in a priority queue. Generally, these operations take $O(|V|\log|V| + |E|)$ time. However, the priorities are bounded by the maximum degree. We use a linked list for each priority class, where each class has the same number of uninformed vertices. This reduces the time complexity to $O(|V| + |E|) = O(|E|)$. Therefore, in each round, the time complexity of TBA is $O(|E|)$ in total.

The pseudocode of the heuristic is presented below. The refinement of TBA is also mentioned in procedure *Calculate_weight* and *Weight*. The refinement will be introduced later in details.

Heuristic *TBA* (*tree based algorithm*) **Input**: graph $G = (V, E)$ and originator u, only u is informed. **Output**: broadcast scheme and $b(u, G)$.

1. round = 0; /* set broadcast time 0 */
2. $bb(round) \leftarrow$ all informed vertices with uninformed neighbors; /* in round 0, only the originator is on the bright border */
3. *Remote* $\leftarrow \emptyset$; /* queue used to calculate the weight of each uninformed vertex */
4. *Uninformed* $\leftarrow |V|$ - 1; /* number of uninformed vertices */
5. **While** Uninformed $\neq 0$
   - 5.1. round = round + 1;
   - 5.2. Perform a variant of BFS from $bb(round)$ to uninformed vertices, and mark any uninformed vertex v with $D(v, round)$;
   - 5.3. For any uninformed vertex v, if v has no children, Remote $\leftarrow v$, v.childrenset $= \emptyset$;
   - 5.4. Procedure Calculate_weight;
   - 5.5. Procedure Calculate_match;
   - 5.6. $bb(round) \leftarrow$ all informed vertices with uninformed neighbors;

Procedure *Calculate_weight*

1. **While** *Remote* is not empty
   - 1.1. $v = Remote$.pop();
   - 1.2. **if** $v.childrenset = \emptyset$;
   - 1.3. **then** $v.weight = 0$;
   - 1.3. /* Refinement version */ ***then*** $v.weight = 1$;
   - 1.4. **else** $v.weight$ = Procedure Weight($v.childrenset$);
   - 1.5. **For** all uninformed neighbors w of v;
     - 1.5.1 **if** $D(w, round) = D(v, round) - 1$;
     - 1.5.2 **then** w.childrenset $\leftarrow v$;
     - 1.5.1 **if** $D(w, round) = D(v, round) - 1$ and w is not in *Remote*;
     - 1.5.4 **then** *Remote*.push(w); /* add w to the end of the queue */

Procedure *Weight* **Input**: $v.childrenset$ **Output**: the weight of vertex v

0. /* Only in refinement version */ ***for*** $w \in v.childrenset$ $w.weight = \frac{(w.weight)\cdot p}{q}$, *where q is the number of parents of w and p is a parameter. /* for each vertex w, this calculation only be performed once in each round although w could have more than one parent.*/*
1. Create *Bucket*[k]; /* k empty buckets, $k =| v.childrenset |$ */
2. $MAX(v) = max\{w.weight \mid w \in v.childrenset\}$;
3. **for** $w \in v.childrenset$
   - 3.1. **if** $MAX(v) - i \geqslant w.weight > MAX(v) - i - 1$, $0 \leqslant i \leqslant k-1$;
   - 3.2. **then** $Bucket[i] \leftarrow w$;
4. **for** $0 \leqslant i \leqslant k-1$
   - 4.1. $SUM(i) = \sum_{j=0}^{i} \mid Bucket(i) \mid$;
   - 4.2. $MIN(i) = min\{w.weight \mid w \in Bucket(i)\}$;
5. **return** $max\{SUM(i) + MIN(i) \mid 0 \leqslant i \leqslant k-1\}$;

Procedure *Calculate_match*

1. list *match*[*degree*]; /* create *degree* lists. *degree* stands for the maximum degree of all vertices in $G = (V, E)$ */
2. **for** all vertices w in $bb(round)$
   - 2.1. *neighbor* = the number of uninformed neighbors of w;
   - 2.2. *match*[*neighbor*-1].add(w);
3. **for** $0 \leqslant i \leqslant degree - 1$
   - 3.1. *match*[i].setcurr();/* set the current pointer in each list point to the first element */
4. **While** not all *current* points in lists of *match*[*degree*] are NULL; and let the first list where *current* is not NULL be *match*[i]
   - 4.1 $w = match[i].getnext()) \neq$ NULL /* get the current element, and assign it to w; *current* points to the next element. */
   - 4.2 v = one of the uninformed neighbors of w with maximum weight;
   - 4.3 Output the broadcast scheme: w sends the message to vertex v in the current round;
   - 4.4 mark v informed and *Uninformed* = *Uninformed* - 1 ;
   - 4.5 **for** all neighbors p of vertex v such that p belongs to a list *match*[j]
     - 4.5.1 **if** $j = 0$ **then** remove p from *match*[j];
     - 4.5.2 **if** $j > 0$ **then** move p from *match*[j] to *match*[j-1]; /* if p was located before the current pointer in *match*[j], then p is also located before the current pointer in *match*[j-1]; and if p was located behind the current pointer in *match*[j], then p is also located behind the current pointer in *match*[j-1] */

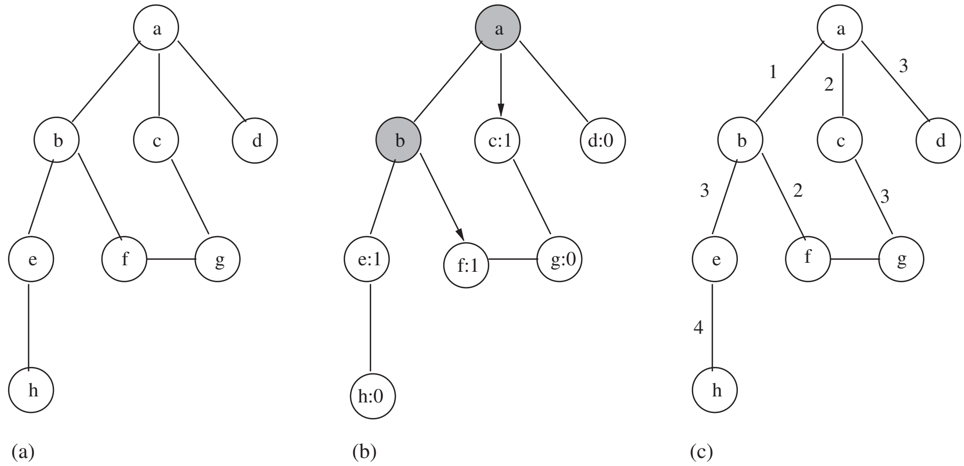

Fig. 2. The performance of TBA.

2.2. Refinement

In TBA, a vertex could a descendant of more than one vertices. Thus, the effect of this vertex on the process of broadcasting is overestimated. Fig. 2 shows such an example. The graph in Fig. 2(a) is the original graph. Vertex a is the originator. The graph in Fig. 2(c) illustrates the broadcast scheme generated by TBA. The weights of each uninformed vertex in round 2 are presented in Fig. 2(b). The vertices with shadowed backgrounds are informed vertices. In the second round, the weights of vertices f and c are equal to 1 because vertex g is a child of both f and c. However, vertex g receives the message either from f or from c, but not from both. Therefore, the effect of vertex g is overestimated. In this example, vertex e and vertex f have the same weight. As a result vertex b could send the message to vertex f in the second round, although sending to vertex e would be a better choice. This motivates the following refinement: dividing the weight of a child by the number of its parents. If a vertex u has no children, then $w(u,t) = 1$. If u has k children $v_1, v_2, \ldots, v_k$, where $w(v_1,t) \geqslant w(v_2,t) \geqslant \cdots \geqslant w(v_k,t)$, then $w(u,t) = max\{\frac{w(v_i,t)\cdot p}{q} + i, 1 \leqslant i \leqslant k\}$. Here q stands for the number of parents of v_i, and p stands for a parameter. For the parameter p, we used integers from 1 to 6. Note that the time complexity of the refinement is the same as that of the original heuristic.

The graph in Fig. 3(a) presents the weights of each uninformed vertex in round 2 by using the refinement. The graph in Fig. 3(b) shows the broadcast scheme generated by the refinement. This is the optimal broadcast scheme from originator a.

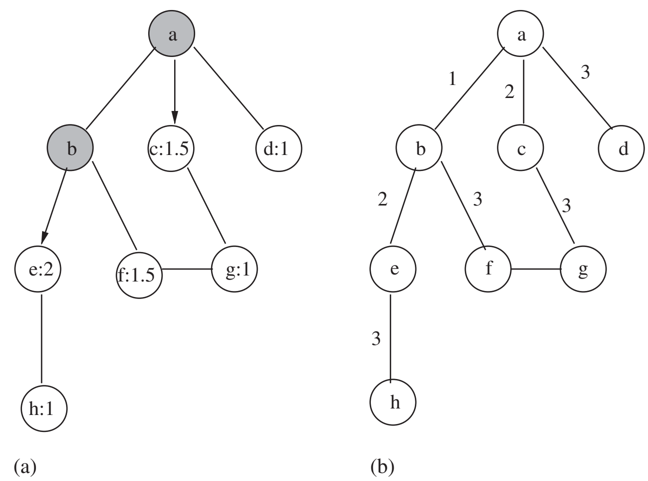

Fig. 3. The performance of the refinement.

3. Theoretical results

It is easy to see that TBA generates the optimal broadcast scheme on several simple topologies, such as *ring* and *tree*. An $m \times n$ grid graph $G_{m,n}$ is the product of path graphs on m and n vertices, while the $m \times n$ torus graph $Torus(m,n)$ is the product of ring graphs on m and n vertices. In grid and torus graphs, the vertical paths or rings are columns, and the horizontal paths or rings are rows. The columns are numbered from 0 to $n-1$. The rows are numbered from 0 to $m-1$. A vertex on the intersection of row i and column j is denoted by (i,j). We will prove that TBA also generates an optimal broadcast scheme on the *grid* graph when the originator is a corner vertex. In fact, we will prove a stronger result: any possible broadcast scheme in a grid is an optimal broadcast scheme if the originator is a corner vertex. We will also show that the upper bound of the broadcast time generated by TBA in the torus is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2 \leqslant b(Torus(m,n)) + 3$. In this chapter, $b(A(u,G))$ stands for the broadcast time of G generated by TBA when the originator is vertex u, and $b(A(G))$ stands for the broadcast time of graph G generated by TBA.

3.1. The grid graph

The following results are presented in [7]: let v be a vertex in $G_{m,n}$, then $b(v, G_{m,n}) = m + n - 2$ when v is a corner vertex. When v is a side vertex, then $b(v, G_{m,n}) =$ the maximum distance from v to a corner vertex plus 1 if there are two corner vertices at the maximum distance and $b(v, G_{m,n}) =$ the maximum distance from v to a corner vertex if there is one corner vertex at the maximum distance. If v is an interior vertex at position (i, j), then $b(v, G_{m,n}) =$ the maximum distance from v to a corner vertex plus 1 if $i = \frac{m-1}{2}$ or $j = \frac{n-1}{2}$, plus 2 if $i = \frac{m-1}{2}$ and $j = \frac{n-1}{2}$, and $b(v, G_{m,n}) =$ the maximum distance from v to a corner vertex otherwise.

Given the originator u and a broadcast scheme S in a graph G, $b(S(u, G))$ stands for the broadcast time of u in graph G by using the broadcast scheme S. In this section, we will prove that for any broadcast scheme S, $b(S((0, 0), G_{m,n})) = b((0, 0), G_{m,n}) = m + n - 2$.

To present the theorem, we need the following definitions: (1) *Border*: the set of informed vertices that have uninformed neighbors. (2) *Outside neighbors* of vertex (i, j): $(i+1, j)$ and $(i, j+1)$. (3) *Convex border*: a border is convex if there are no two vertices (i, j) and (p, q) on the border such that $i > p$ and $j > q$.

Fig. 4 illustrates the above definitions. The vertices with black backgrounds are informed vertices, and the vertices with white backgrounds are uninformed vertices. The vertices connected by bold edges compose the *border*. Vertices P and Q are the outside neighbors of vertex O. The border in Fig. 4(a) is convex, while the border in Fig. 4(b) is not convex, since vertices $A = (2, 2)$ and $B = (3, 4)$ are on the border and $2 < 3, 2 < 4$. First, we will prove some auxiliary lemmas.

Lemma 1. *Given a vertex (i, j) on a convex border, any other vertex (p, q), where $0 \leqslant p \leqslant i$ and $0 \leqslant q \leqslant j$, is informed.*

Proof. Assume that there exist uninformed vertices (p, q), where $0 \leqslant p \leqslant i$ and $0 \leqslant q \leqslant j$. Consider any such vertex (x, y) that has the shortest distance from $(0, 0)$. This means that both $(x - 1, y)$ and $(x, y - 1)$ are informed. Then both $(x, y - 1)$ and $(x - 1, y)$ are on the convex border. If $x < i$ and $y \leqslant j$, then this is a contradiction, since $x < i$ and $y - 1 < j$. If $x \leqslant i$ and $y < j$, then this is also a contradiction, since $x - 1 < i$ and $y < j$. □

Lemma 2. *The border is convex after each round of any broadcast scheme originated by vertex* $(0, 0)$.

Proof. We will prove this lemma by induction on the number of rounds. At the beginning, $(0, 0)$ is the only informed vertex. In the first round, $(0, 0)$ informs either $(0, 1)$ or $(1, 0)$, which generates a convex border.

Assume that after round t, the border generated by any broadcast scheme S is convex. We should prove that the border will be convex after round $t + 1$. Assume that the border is not convex after round $t + 1$. Then, there exist vertices (i, j) and (p, q) on the border, where $i < p$ and $j < q$. It is easy to see that (p, q) was not on the border after round t, since either (i, j) or one of $(i - 1, j)$ and $(i, j - 1)$ were on the convex border after round t. Thus, vertex (p, q) received the message at time $t + 1$ either from $(p - 1, q)$ or $(p, q - 1)$. So, at least one of $(p - 1, q)$ and $(p, q - 1)$ was on the convex border after round t. Consider the case that $(p - 1, q)$ was on the convex border after round t. By Lemma 1, after round t, vertex (i, j) was informed (since $i \leqslant p - 1$ and $j < q$) and it had at most one uninformed neighbor, $(i + 1, j)$. So, after round $t + 1$, $(i + 1, j)$ is informed, and (i, j) cannot be on the border since it has no uninformed neighbors. This contradicts the assumption of the lemma. Similarly, we can get a contradiction for the case that $(p, q - 1)$ was on the convex border after round t. □

Theorem 1. $b(S((0, 0), G_{m,n})) = m + n - 2$.

Proof. From Lemma 1, any vertex on a convex border can only send the message to its outside neighbors. From Lemmas 1 and 2, the longest distance between the vertices on the border and the originator increases by 1 at each round. The vertex $(m - 1, n - 1)$ is informed in round $m + n - 2$ since it is the only vertex in the grid that has distance $m + n - 2$ from $(0, 0)$. From Lemma 1, after vertex $(m - 1, n - 1)$ is informed, then the broadcasting is complete. Thus, $b(S((0, 0), G_{m,n})) = m + n - 2$. □

The above theorem proves that the broadcast time of any broadcast scheme from originator $(0, 0)$ is equal to the diameter of the grid. The broadcast time of TBA follows directly.

Theorem 2. $b(A((0, 0), G_{m,n})) = m + n - 2$.

It is natural to state that $b(A((x, y), G_{m,n})) \leqslant m+n-2$ for any originator (x, y). All the testing results of TBA confirm the above statement (see Table 1). In this table, the originator is listed in the columns labeled by O, and the broadcast times are listed in the column labeled by R. Moreover, TBA always generates the theoretical minimum broadcast time (see [7]) from all originators. However, we were unable to prove the above statement mathematically.

3.2. The Torus graph

TBA generates almost optimal broadcast time in the torus. The optimal broadcast time of Torus(m, n) is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 1$ when both m and n are odd, and is $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil$ otherwise [9]. In this section, we will show that $b(A(Torus(m, n))) \leqslant \lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2 \leqslant b(Torus(m, n)) + 3$.

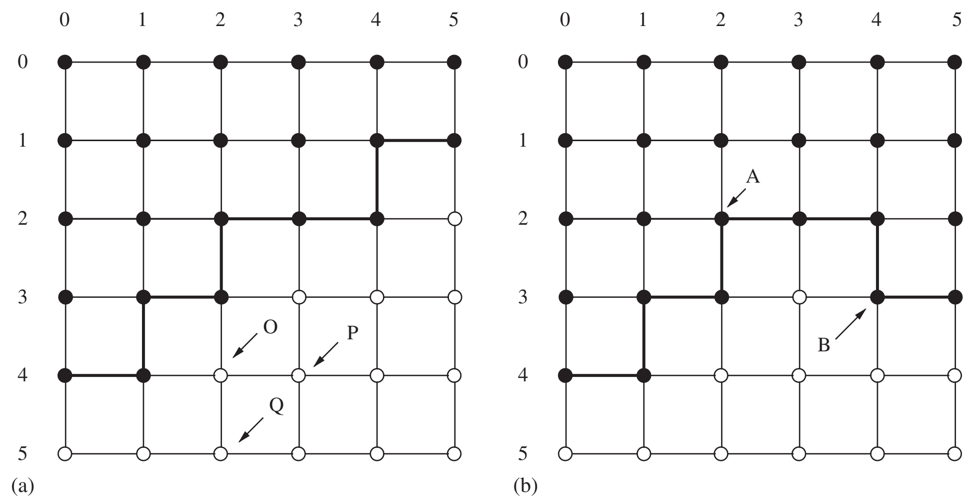

Fig. 4. Definitions in the grid graph.

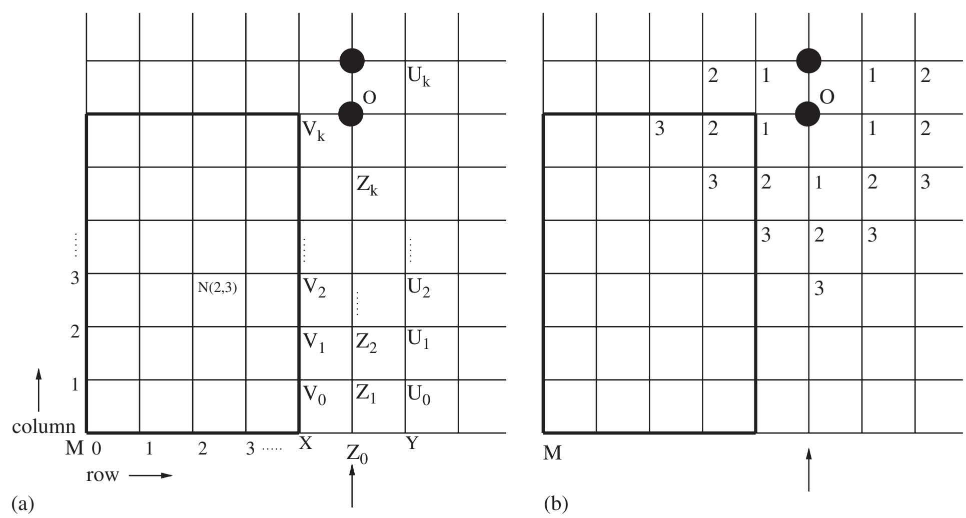

Fig. 5. The two-dimensional Torus graph.

Table 1
Testing results in $G_{m,n}$

$G_{20,30}$		$G_{50,30}$		$G_{15,25}$		$G_{20,25}$	
O	R	O	R	O	R	O	R
0,0	48	0,0	78	0,0	38	0,0	43
3,2	43	9,6	63	3,5	30	3,2	38
5,3	40	12,7	59	6,8	24	5,8	30
9,5	34	12,14	52	7,10	22	8,4	31
10,7	32	15,20	54	7,12	21	10,12	23
11,9	31	15,25	59	9,15	24	15,10	29
10,15	25	20,25	54	11,16	27	15,16	31
15,10	34	25,15	40	12,20	32	18,20	38
15,20	35	30,18	48	12,22	34	18,24	42
19,28	47	45,28	73	14,22	36	12,24	36

Without loss of generality we assume that in round one the originator sends the message to a neighbor in the same column. Fig. 5 illustrates $Torus(m, n)$ after the first round. Fig. 5(b) shows the distance between vertices in the torus and the originator vertex O. The vertices with solid background are informed vertices, and vertex M is the uninformed vertex that has the longest distance from vertex O. Every vertex in the area defined by thick lines has two children except vertices in row 0 or in column 0. Parent–child relationship is defined as in TBA (Section 2.1). The vertices in the column indicated by arrow have three children except vertex Z_0 (which has two children). The two children of (i, j) are $(i-1, j)$ and $(j-1, i)$. Vertex $(0, j)$ has one child $(0, j-1)$ for $j > 0$. Vertex $(i, 0)$ has one child $(i-1, 0)$ for $i > 0$. Vertex $(0, 0)$ does not have any children because it has the longest distance from vertex O. The weight of a vertex $N = (i, j)$ is denoted by $w(N)$ or $w(i, j)$.

Before proving the main theorem, we first present some auxiliary lemmas.

Lemma 3. *In the area defined by thick lines*, $w(i, j) = i + j + min\{i, j\}$.

Proof. Lemma 3 can be proved by induction. The statement is correct for vertex $(0, 0)$ since $w(0, 0) = 0 + 0 + min\{0, 0\} = 0$. For all the vertices that are on row 0, assume that $w(0, j) = 0 + j + min\{0, j\} = j$, then $w(0, j + 1) = w(0, j) + 1 = j + 1 = 0 + (j + 1) + min\{0, j + 1\}$. For the vertices that are on column 0, the proof is similar. Assume that the statement is correct for all descendants of (i, j) ($i \neq 0$ and $j \neq 0$). Vertex (i, j) has two children $(i - 1, j)$ and $(i, j-1)$. If $i > j$, then $min\{i-1, j\} = j$ and $min\{i, j-1\} = j-1$. So, $w(i-1, j) = i-1+j+min\{i-1, j\} = i+2j-1$ and $w(i, j-1) = i+j-1+min\{i, j-1\} = i+2j-2$. Then, $w(i, j) = w(i - 1, j) + 1 = i + 2j = i + j + min\{i, j\}$. If $i < j$, then $min\{i - 1, j\} = i - 1$ and $min\{i, j - 1\} = i$. So, $w(i, j-1) = 2i + j - 1$ and $w(i-1, j) = 2i + j - 2$. Then, $w(i, j) = w(i, j - 1) + 1 = 2i + j = i + j + min\{i, j\}$. If $i = j$, then $w(i, j - 1) = i + 2j - 2 = w(i - 1, j)$. So, $w(i, j) = w(i-1, j)+2 = i+2j = i+j+min\{i, j\}$. □

Lemma 4. $w(Z_0) \geqslant w(V_0) = w(U_0)$.

Proof. Let $Z_0 = (0, p)$. By Lemma 3, $w(X) = p-1$. Similarly, $w(Y) = p - 1$. Since X and Y are two children of Z_0, then $w(Z_0) = p+1$. By Lemma 3, $w(V_0) = w(p-1, 1) = p+min\{p-1, 1\}$. $w(Z_0)-w(V_0) = 1-min\{p-1, 1\}$. When $p - 1 \geqslant 1$, then $w(Z_0) - w(V_0) = 0$, and when $p - 1 < 1$, then $w(Z_0) - w(V_0) > 0$. Therefore, $w(Z_0) \geqslant w(V_0)$. The proof of $w(V_0) = w(U_0)$ is simple. □

Lemma 5. $w(Z_i) > w(V_i) = w(U_i)$, *for* $i = 1, 2, \ldots, k$.

Proof. TBA assigns the same weights to vertices V_i and U_i. So, $w(V_i) = w(U_i)$, for $i = 1, 2, \ldots, k$.

By Lemma 4, $w(Z_0) \geqslant w(V_0) = w(U_0)$. Z_1 has three children: Z_0, V_0 and U_0. By the definition of the weight, $w(Z_1) \geqslant w(V_0)+3$. Let $V_0 = (p, q)$, then $w(Z_1) \geqslant w(V_0) + 3 = p + q + min\{p, q\} + 3$. $w(V_1) = w(p, q + 1) = p + q + 1 + min\{p, q + 1\}$. $w(Z_1) - w(V_1) \geqslant min\{p, q\} + 2 - min\{p, q + 1\}$. When $p \leqslant q$, $w(Z_1) - w(V_1) \geqslant 2$. When $p > q$, $w(Z_1) - w(V_1) \geqslant 1$. So, $w(Z_1) > w(V_1) = w(U_1)$. Assuming $w(Z_i) > w(V_i) = w(U_i)$ and $V_i = (p, q)$, we have $w(Z_{i+1}) \geqslant w(V_i) + 3 = p + q + min\{p, q\} + 3$. $w(V_{i+1}) = w(p, q + 1) = p + q + 1 + min\{p, q + 1\}$. So, $w(Z_{i+1}) - w(V_{i+1}) \geqslant min\{p, q\} + 2 - min\{p, q + 1\} > 0$. Thus, $w(Z_{i+1}) > w(V_{i+1}) = w(U_{i+1})$. Therefore, $w(Z_i) > w(V_i) = w(U_i)$, for $1 \leqslant i \leqslant k$. □

Theorem 3. $b(A(Torus(m, n))) \leqslant \lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2$.

Proof. By Lemma 5, $w(Z_i) > w(V_i) = w(U_i)$, for $1 \leqslant i \leqslant k$. Thus, vertex Z_i receives the message before vertices V_i and U_i for any $i = k, k - 1, \ldots, 1$. Since Z_0 is the furthest vertex from the originator on the same column, then Z_1 receives the message at round $\lceil \frac{m}{2} \rceil - 1$. After this, it is possible that Z_1 sends to V_0 or U_0 first, because $w(Z_0) \geqslant w(V_0) = w(U_0)$. In the worst case, Z_0 first sends to V_0, then U_0, and finally sends to Z_0. This takes 3 rounds. After this, vertices $Z_0, Z_1, \ldots, Z_k$ and all the other vertices on the same column are informed. It takes at most $\lceil \frac{n}{2} \rceil$ rounds more to finish the broadcasting using horizontal edges. Thus, $b(A(Torus(m, n))) \leqslant \lceil \frac{m}{2} \rceil - 1 + 3 + \lceil \frac{n}{2} \rceil = \lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil + 2$. □

4. Experimental results

In this section, the testing results in several commonly used topologies and three graph models are presented.

4.1. Testing results in commonly used topologies

In this section and the following tables (Tables 2–4), S_d, BF_d, H_d, SE_d, $UB(2, d)$ and CCC_d abbreviate the *Star* graph, *Butterfly* graph, *HyperCube* graph, *Shuffle Exchange* graph, *deBruijn* graph and *Cube-Connected Cycle* of dimension d in the stated order. *Low* and *Up* stands for the best known theoretical lower and upper bounds, respectively. *TB* stands for the minimum testing result of TBA. *Opt* is the optimal broadcast time of a graph. For these bounds and optimal broadcast times we refer to [3,4,9,13,16]. As in [2], we tested TBA on $UB(2, d)$, SE_d, BF_d and CCC_d. We were able to run it for $d \leqslant 20$ in $UB(2, d)$ and SE_d, and for $d \leqslant 16$ in BF_d and CCC_d, while in [2], the authors have values for $d \leqslant 14$. All our results for $d \leqslant 14$ are the same as in [2], except for 4 cases: in 2 cases our algorithm gives better result (mentioned by *) and in 2 cases their result is better (mentioned by $^-$). In all the four cases the difference is one round. In fact, we generated new upper bounds on broadcast time for CCC_d when $d = 15$, for BF_d when $15 \leqslant d \leqslant 16$, and for $UB(2, d)$ and SE_d when $15 \leqslant d \leqslant 20$. In addition, we tested TBA on H_d and S_d graph, generating again new upper bounds for S_d.

4.2. Testing results in three graph models

ns-2 is a widely used simulator for networking research, which creates topologies by using several models. To compare TBA with the algorithm from [2], three different network models from ns-2 are considered: GT-ITM *Pure Random* [20], GT-ITM *Transit-Stub* (TS) [20] and *Tiers* [5].

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

Table 2
Testing results in CCC_d and BF_d

d	CCC_d			BF_d		
	Low	*Up*	*TB*	*Low*	*Up*	*TB*
3	6	7	6	5	5	5
4	9	9	9	7	7	7
5	11	12	11	8	9	9
6	13	14	13	10	11	10
7	16	17	16	11	13	12
8	18	19	18	13	15	14
9	21	22	21	15	17	16
10	23	24	23	16	19	18^-
11	26	27	26	18	21	19
12	28	29	28	19	23	21*
13	31	32	31	21	25	23
14	33	34	33	23	27	25^-
15	36	37	36	24	29	27
16	38	39	39	26	31	29

Table 3
Testing results in S_d

	S_d						
d	*Low*	*Up*	*TB*	d	*Low*	*Up*	*TB*
3	3	3	3	7	13	16	14
4	5	6	5	8	16	21	16
5	7	9	8	9	19	22	20
6	9	13	11				

Table 4
Testing results in H_d, $UB(2,d)$ and SE_d

d	H_d		$UB(2,d)$			SE_d	
	Opt	*TB*	*Low*	*Up*	*TB*	*Opt*	*TB*
5	5	5	7	9	6*	9	9
6	6	6	8	11	8	11	11
7	7	7	10	12	9	13	13
8	8	9	11	14	11	15	15
9	9	10	12	15	12	17	17
10	10	11	14	17	14	19	19
11	11	12	15	18	15	21	21
12	12	13	16	20	17	23	24
13	13	14	18	21	18	25	26
14	14	15	19	23	20	27	28
15	15	16	20	24	21	29	30
16	16	17	22	26	23	31	32
17	17	18	23	27	25	33	34
18	18	19	24	29	26	35	36
19	19	20	26	30	28	37	38
20	20	21	27	32	29	39	40

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.

Table 5
Testing results in Tiers model: 1105 vertices

Tiers: 1105 vertices					
Edges	*RH*	*TB*	Edges	*RH*	TB
1106	24	24	1324	23	21
1110	24	23	1326	23	21
1214	22	21	1331	20	20
1216	22	21	1447	22	21
1220	22	21	1449	21	20

Table 6
Testing results in Tiers model: 2210 vertices

Tiers: 2210 vertices					
Edges	*RH*	*TB*	Edges	*RH*	*TB*
2209	28	27	3028	31	29
2234	26	25	3209	30	29
2409	32	31	3225	26	24
2427	25	24	3409	32	32
2609	33	32	3428	27	26
2628	26	26	3609	30	29
2809	29	29	3627	30	29
2833	27	27	3809	28	28
3009	32	31	4207	27	26

Table 7
Testing results in GT-ITM Pure Random model

Pure random							
Vertices	Edges	*RH*	*TB*	Vertices	Edges	*RH*	*TB*
200	346	10	10	500	1725	10	10
200	475	9	8	500	1830	10	9
200	595	8	8	750	2099	11	11
300	684	10	10	750	2236	11	10
300	756	10	9				

Table 8
Testing results in TS model: 600 vertices

TS: 600 vertices					
Edges	*RH*	*TB*	Edges	*RH*	*TB*
1169	14	13	1222	15	14
1190	14	14	1231	14	13
1200	16	15	1232	14	13
1206	14	14	1247*	13	14
1219	15	14	1280	14	13

The tables in this section represent some of the testing results of TBA and the algorithm from [2] in the above three models. The results of the algorithm from [2] and TBA are presented in column *RH* and *TB*, respectively. In total we considered about 200 different graphs using the

Table 9
Testing results in TS model: 1056 vertices

TS: 1056 vertices					
Edges	*RH*	*TB*	Edges	*RH*	*TB*
2115	17	16	2176	17	16
2121	17	17	2177	18	17
2134	17	16	2185	16	16
2142	16	15	2187	16	15
2147	16	15	2204	16	15
2149	16	15	2219	17	16
2151	15	15	2220	15	15
2167	17	16	2230	16	15
2169	17	17	2255	15	14

above three models for $155 \leqslant |V| \leqslant 4400$. In only one case (shown by * in Table 8), TBA gave a broadcast time that was one more than the broadcast time obtained by using the algorithm from [2]. In all other cases, we obtained either the same broadcast time as in [2] or better. In the Pure Random model we got a 12% improvement. In the Transit-Stub model TBA gave better broadcast time in more than 40% of the cases. TBA worked better under the Tiers model, as it gave a smaller broadcast time in about 60% of the cases.

Tables 5 and 6 present the testing results in the Tiers model. We present the testing results in the GT-ITM Pure Random model in Table 7, and the testing results in the GT-ITM TS model in Tables 8 and 9.

Acknowledgments

We gratefully acknowledge the insightful comments and suggestions made by the anonymous referees, which helped us significantly improve the presentation of the paper.

References

[2] R. Beier, J.F. Sibeyn, A powerful heuristic for telephone gossiping, in: The Seventh International Colloquium on Structural Information & Communication Complexity (SIROCCO 2000), L'Aquila, Italy, 2000, pp. 17–36.
[3] P. Berthomé, A. Ferreira, S. Perennes, Optimal information dissemination in star and pancake networks, IEEE Trans. Parallel Distributed Systems 7 (12) (1996) 1292–1300.
[4] S. Djelloul, Etudes de Certains Réseaux d'Interconnexion: Structures et Communications, Ph.D. Thesis, Université Paris-Sud, Orsay, 1992.
[5] M.B. Doar, A better model for generating test networks, in: IEEE GLOBECOM'96, London, UK, 1996.
[6] M. Elkin, G. Kortsarz, Sublogarithmic approximation for telephone multicast: path out of jungle, in: Symposium on Discrete Algorithms, Baltimore, MA, 2003, pp. 76–85.
[7] A. Farley, S. Hedetniemi, Broadcasting in grid graphs, in: Proceedings of the Nineth SE Conference on Combinatorics, Graph Theory and Computing, Utilitas Mathematica, Winnipeg, 1978, pp. 275–288.
[8] U. Feige, D. Peleg, P. Raghavan, E. Upfal, Randomized broadcast in networks, in: SIGAL International Symposium on Algorithms, 1990, pp. 128–137.
[9] P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks, Discrete Appl. Math. 53 (1994) 79–133.
[10] P. Fraigniaud, S. Vial, Approximation algorithms for broadcasting and gossiping, J. Parallel Distributed Comput. 43 (1) (1997) 47–55.
[13] R. Klasing, B. Monien, R. Peine, E.A. Stöhr, Broadcasting in butterfly and deBruijn networks, Discrete Appl. Math. 53 (1994) 183–197.
[14] G. Kortsarz, D. Peleg, Approximation algorithms for minimum time broadcast, SIAM J. Discrete Math. 8 (1995) 401–427.
[16] A.L. Liestman, J.G. Peters, Broadcast networks of bounded degree, SIAM J. Discrete Math. 1 (4) (1988) 531–540.
[17] R. Ravi, Rapid rumor ramification: approximating the minimum broadcast time, in: 35th Symposium on Foundation of Computer Science, 1994, pp. 202–213.
[18] P. Scheuerman, G. Wu, Heuristic Algorithms for Broadcasting in Point-to-Point Computer Network, IEEE Trans. Comput. C-33 (9) (1984) 804–811.
[19] P.J. Slater, E.J. Cockayne, S.T. Heditniemi, Information dissemination in trees, SIAM J. Comput. 10 (4) (1981) 692–701.
[20] E.W. Zegura, K. Calvert, S. Bhattacharjee, How to model an internetwork, in: IEEE INFOCOM, San Francisco, CA, 1996.

Further reading

[1] W. Aiello, F. Chung, L. Lu, Random evolution in massive graphs, in: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 510–519.
[11] S.M. Hedetniemi, S.T. Hedetniemi, A.L. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1988) 319–349.
[12] J. Hromkovic, R. Klasing, B. Monien, R. Peine, Dissemination of information in interconnection networks, in: D.-Z. Du, D.F. Hsu (Eds.), Combinatorial Network Theory, Kluwer Academic Publishers, Dordrecht, 1996, pp. 125–212.
[15] T. Leighton, Introduction to Parallel Algorithms and Architectures: Array-Trees-Hypercubes, Morgan-Kaufmann Publishers, San Mateo, CA, 1992.

Hovhannes Harutyunyan is an associate professor in the Department of Computer Science and Software Engineering at Concordia University. He received his Bachelor and Master in Applied Mathematics from Yerevan State University, Armenia, and his Ph.D in Mathematics from the Armenian Academy of Sciences. Before Concordia, Dr. Harutyunyan worked at Armenian Academy of Sciences, Simon Fraser University and at Brandon University. He was a researcher at Swiss Federal Institute of Technology (ETH), Zurich and Bielefeld University, Germany. His main research area includes message dissemination problems in networks. He is also interested in graph theory, design and analysis of algorithms, error-correcting codes, cryptography, diagnosis of computer networks.

Bin Shao is currently completing his Ph.D. in computer science under the supervision of Dr. Harutyunyan at Concordia University. He is working on efficient algorithms and network topologies for information dissemination. He received his Bachelor of Engineering from Beijing University of Chemical Technology in 1997, after which he had worked on information management and data compression in the Chinese Academy of Science until 2000. He received his Master degree in Computer Science at Concordia University in 2003 for work on a heuristic of broadcasting. His interests include graph theory and algorithm design.