Broadcasting on Anonymous Unoriented Tori*

Stefan Dobrev and Peter Ružička

Institute of Informatics
Comenius University
Slovakia
E–mail:{dobrev,ruzicka}@dcs.fmph.uniba.sk

Abstract. We consider broadcasting on asynchronous anonymous totally unoriented $n \times m$ torus, where $N = n \cdot m$ is the number of nodes. We present a broadcasting algorithm with message complexity $1.43N + O(n + m)$ and prove the lower bound in the form $1.14N - O(1)$. This is an improvement over the previous $2N + O(\sqrt{N})$ upper bound and $1.04N - O(\sqrt{N})$ lower bound achieved by Diks, Kranakis and Pelc [DKP96]. Unlike the algorithm from [DKP96], our algorithm works also on non-square tori, does not require the knowledge of sizes n and m and uses only messages of size $O(1)$ bits. This is the first known broadcasting algorithm on unoriented tori that does not use all edges.

1 Introduction

Broadcasting is one of the most fundamental communication tasks in parallel and distributed computing. One node of the network, called *source*, has a message which has to be transmitted to all other nodes in the network.

The complexity of broadcasting strongly depends on the amount of topological information available at nodes. If links of a network are globally consistently labelled, forming *sense of direction* [FMS95,Tel95a], broadcasting is possible using only linear number of messages w.r.t. the number of nodes [FMS96]. But if a network is unoriented (no consistent global labels on links are available), then the lower bound for general networks is linear in the number of links [FMS96]. This lower bound is achievable by the naive broadcasting algorithm, in which a node immediately spreads the message to all neighbours except to the one from which it received it.

While this strategy cannot be improved on general networks, broadcasting algorithms might exploit the knowledge of special topologies to reduce the number of messages. For example, on the complete unoriented N–node network the broadcasting is trivially accomplished by sending only $N - 1$ messages from the source to all its neighbours. Another, not so trivial example, is the class of unoriented N–node chordal ring networks with chords leading to $2k$ closest neighbours in the ring, where the broadcasting can be performed using only $O(N)$ messages [Pel97].

* The research was partially supported by EU Grant No. INCO-COP 96-0195 "ALTEC-KIT" and by the Slovak VEGA project 1/4315/97.

J. Hromkovič, O. Sýkora (Eds.): WG'98, LNCS 1517, pp. 50–62, 1998.

Other results have been obtained for broadcasting on special topologies without orientation. It was stated as an open question (see [Tel95a], cf. also [Mans96]) whether there exists a broadcasting algorithm on unoriented N–node hypercubes using only $O(N)$ messages in the worst case. This question was positively answered only recently in [DR97,DKP96,DDKPR98], where two different independently obtained linear message algorithms for broadcasting on unoriented anonymous hypercubes were presented. Further improvements of these results in the time and bit complexity can be found in [DRT98].

We are interested in broadcasting on unoriented tori. Since tori have constant degree, the naive broadcasting algorithm using $3N+1$ messages is asymptotically optimal. However, it is possible to improve the constant factor, as documented by the algorithm from [DKP96], which uses $2N+O(\sqrt{N})$ messages on $\sqrt{N}\times\sqrt{N}$ tori. In [DKP96] it was also shown that the lower bound is non-trivial ($1.04N - O(\sqrt{N})$ messages). We further improve these results, both in upper and lower bounds. We present $1.43N + O(n + m)$ message broadcasting algorithm which, unlike the algorithm from [DKP96], works also on non-square torus, does not need to know its size and uses only messages of size $O(1)$. This is the first know broadcasting algorithm working on unlabeled tori that does not use every link. We also present an improved lower bound $1.14N - O(1)$, using a technique with potential for further improvement of this lower bound. Our lower bound is on the number of used links and it applies also for the case of synchronous communication, regardless whether vertices know the size of torus.

The paper is organized as follows. In Section 2 we present the computational model. In Section 3 we show $1.43N + O(n + m)$ broadcasting algorithm. In Section 4 we prove an $1.14N - O(1)$ lower bound for broadcasting on unlabeled N–node tori.

2 The Model

The computational model is the standard model of asynchronous distributed computing [Tel94b]. Every message will be delivered in a finite but unbounded time. FIFO on links is not required. All processors are identical and run the same algorithm.

The underlying topology of the network is anonymous unoriented torus of size $n \times m$. Anonymity means that processors do not have given distinct identities. Unorientation means that each processor can distinguish its links only by uninterpreted labels 1, 2, 3 and 4. However, this labeling is arbitrary at each processor and labels are thus without any topological meaning. That means that if a processor sends a message on an unused link, the actual link (from the set of yet unused links) is chosen by the adversary, as all unused links look alike to the sender and we are interested in the worst case behaviour.

We are interested in *communication complexity*, expressed by the number of messages sent in the worst case. The worst case refers to the adversary decisions concerning choices of yet unused links, and to the worst possible message delays.

We are considering the problem of *broadcasting*. At the beginning there is a single active processor – the source of information. Other processors will became active only after receiving a message. Only active processors can send messages. At the end of computation we require each processor to be active (it has received an information).

3 Upper Bound

In this section we present a broadcasting algorithm on asynchronous anonymous totally unoriented $n \times m$ tori using $\frac{10}{7}n \cdot m + O(n+m)$ messages in the worst case.

3.1 Informal Description of the Algorithm

Our algorithm $\mathcal{A}$ starts from the source s by sending an initial message in one direction until it returns back to s. This can be done using *handrail* technique from [Pet85,Mans96b] with $O(n)$ messages of size $O(1)$, where n is the size of the torus in the direction of the initial message. Created path circling around tori is called *equator*. From the handrail technique follows that vertices on the equator can consistently distinguish between their *north* and *south* sides.

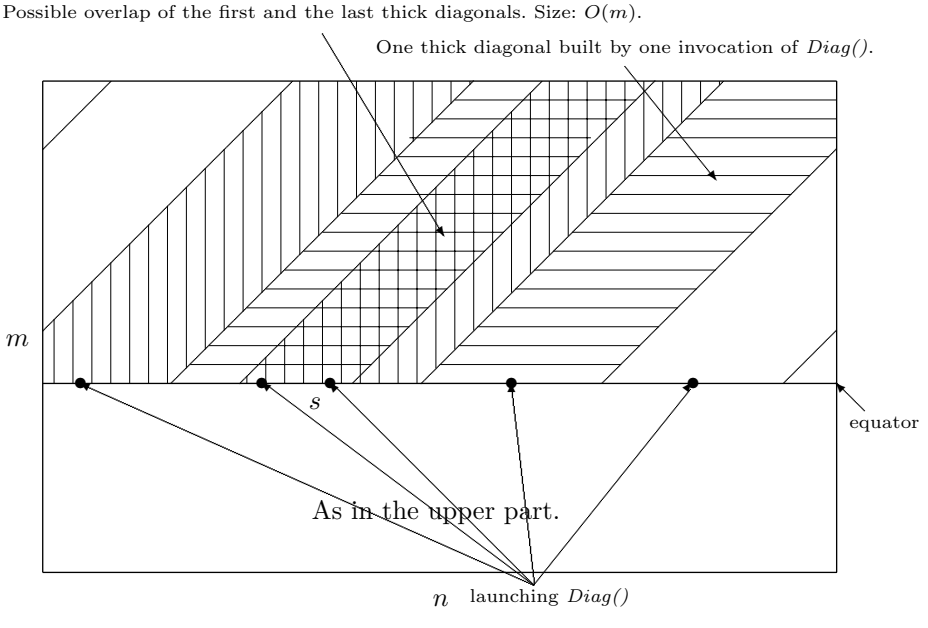

Figure 1.

In the second phase, another message is sent along the equator eastward and at each 7^{th} vertex it launches northward the subroutine *Diag()* until it returns back to the source. The first launch of *Diag()*, denoted as $Diag^1()$, is done using

marked messages, so it will not interfere with the last one, which may overlap with it. *Diag()* procedure broadcasts on thick (7 vertex) diagonal in north-east direction until it returns back to the equator.

The overall message complexity of the broadcasting algorithm is $\frac{10}{7}N + O(n+m)$ and it follows from these facts:

- The cost of the start-up (building the equator and launching *Diag()* procedures) is $O(n)$.
- *Diag()* uses $\frac{10}{7}$ messages of size $O(1)$ bits per each reached vertex.
- The whole torus can be covered by disjoint thick diagonals built by *Diag()*. Since *Diag()* stops on the equator, different invocations of *Diag()* do not overlap. The only exception is the first and the last invocation of *Diag()*, which can overlap for n non-multiple of 7, where n is the length of the equator. There are at most $6m$ vertices in the intersection, that means totally $\frac{60}{7}m$ additional messages. (See Figure 1)

3.2 Detailed Description of the Algorithm

StartUp

At the source on start up:

0. For each incident link h:
   Send(S_x^1), where x is label of h at the source vertex;

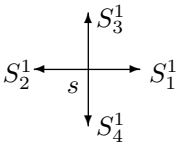

Figure 2.

At arbitrary vertex:

	Upon receiving	Send	See Figure
1.	S_x^1	S_x^2 on all remaining links	3
2.	S_1^2 on h and S_x^2	S_x^3 on h	4
3.	S_x^3 on h_1 and S_y^3 on h_2, let $x < y$	U_1 on h_1, B_1 on h_2 and M_1 on link on which S_1^2 was sent, but nothing received.	5

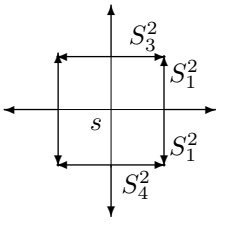

Figure 3.

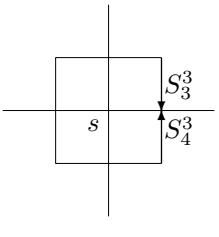

Figure 4.

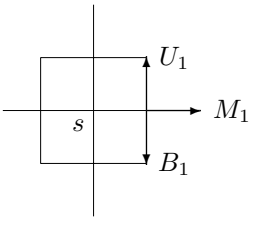

Figure 5.

Building the equator:

	After receiving message(s)	Send	See Figure
4a.	U_1	U on unused links	6
4b.	B_1	B on unused links	6
4c.	M_1, not at source	M on unused links [1]	5,6 and 9
4d.	M_1 at source	L on link with label 1 [2]	10
5a.	U on h and M	U on all links except h	7
5b.	B on h and M	B on all links except h	7
5c.	U and B	M_1 on links on which no M_1, U or B was received	8

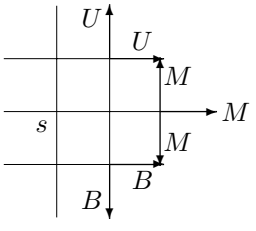

Figure 6.

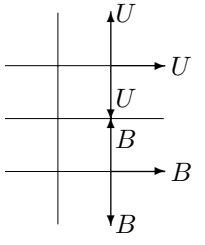

Figure 7.

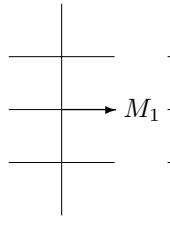

Figure 8.

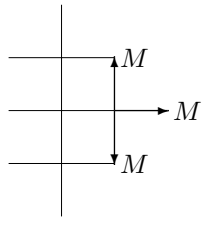

Figure 9.

Launching phase

	After receiving message(s)	Send	See Figure
6.	L	D_0' where U_1 was sent	5 and 10
		L_1 where M_1 was sent	
7.	L_i, not at source	$L_{(i+1) \bmod 7}$ where M_1 was sent	8 and 10
		if $i = 0$, send D_0 from where U message came	7 and 10

Diag() procedure:

	After receiving message(s)	Send to all unused links [3]
8a.	D_0	D_1 [4]
8b.	D_1 and not M received before	D_2
8c.	D_2 and (D_9 or S_x^2)	D_3
8d.	Two D_2	D_4
8e.	D_4	D_5
8f.	Two D_5	D_6
8g.	D_6	D_7
8h.	D_5 and D_7	D_3
8i.	Two D_7	D_8
8j.	D_8	D_9
8k.	Two D_9	D_1

[1] Links used only by S_x^y messages are considered unused.
[2] Starting launching phase.

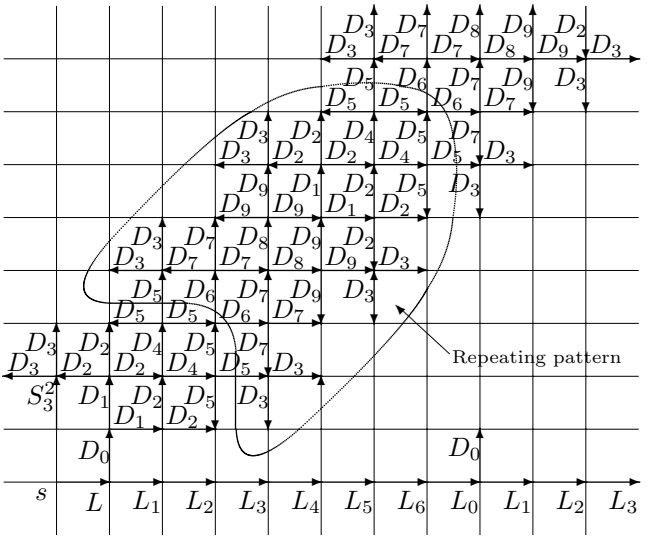

Figure 10.

If D'_0 message is received, D'_i messages will be used, to avoid possible interference between the first and the last invocation of *Diag()*.

It is easy to see that computation cyclically proceeds in cycle 8a→8b→8d→ →8e→8f→8g→8i→8j→8k→8b with concurrent steps 8c and 8h. In one such cycle altogether 28 new vertices are added to the thick diagonal using 40 messages, resulting in overall $\frac{10}{7}$ messages per vertex.

Diag() terminates when it returns back to the equator from the south – a vertex that has received M and B message will not send any *Diag()* message.

3.3 Making Termination Explicit

The algorithm presented in subsection 3.2 terminates implicitly. One way to make the termination explicit within the same complexity bound is the following:

- Vertices reached by messages of *Diag()* terminate when they finish their work in Diag().
- Vertices of the equator terminate when the launching token of the second phase has passed them.
- The only problem are vertices of the first – $Diag^1()$ and the last – $Diag^q()$ thick diagonal. The problem is how to terminate in order to nonblock broadcasting on the second thick diagonal. One possible solution is the following:
  - When $Diag^1()$ returns back to the equator, it returns k steps to the west and launches $Diag^q()$ to south-west. Vertices reached by $Diag^q()$

[3] Unused links – unused by D_i and M messages. No messages are sent from vertices that received a B and M message – termination of *Diag()* when equator was reached from the south.

[4] Messages are sent along two links on which no U, D_0 or S_x^y message arrived. (see Figure 6)

but not by $Diag^1()$ will terminate after finishing their work in $Diag^q()$. When $Diag^q()$ reaches the equator, it goes k steps eastward and launches $Diag^1()$ to north-east. Now vertices can terminate after finishing their work in $Diag^1()$. k can be computed during the construction of the equator as the length of the equator modulo 7. This additional computation can be done using $O(m)$ messages.

4 Lower Bound

We will prove the lower bound by letting the algorithm and the adversary play the following game:

- At the beginning the domain D of the algorithm consists of a single vertex – the source.
- The goal of the algorithm is to extend its domain to the whole torus.
- The game proceeds in rounds. Each round begins with a move of the algorithm, which is followed by a move of the adversary.
- The algorithm knows the graph D representing its domain, but does not know how it is embedded into torus.
- The algorithm specifies (during its move) for each vertex of its domain the number of yet unexplored links it wants to explore.
- The adversary chooses an embedding of D into torus and decides which of yet unexplored links at given vertex will be explored. (According to the orders given by the algorithm.)
- At the end of each round all explored links are added to the domain of the algorithm.
- The game terminates when the domain spans the whole torus.
- The goal of the adversary is to maximize the number of explored links.

Since the game proceeds in synchronous rounds, the lower bound applies also for the case of synchronous communication. Moreover, this is the lower bound on the number of used links, not only on the number of used messages.

Our adversary tests all possible embeddings, and for each embedding it tests all possible ways of choosing explored links. The embedding (and the choice of explored links) which leads to the smallest new domain is chosen. (See Figure 11)

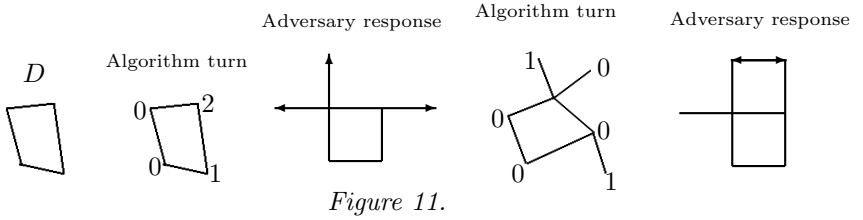

Figure 11.

Let D_i be the domain of the algorithm after the round i. D_i can be divided to the *core graph* C_i and *hanging trees* T_i. The *core graph* can be defined as the

maximal subgraph C_i of D_i such that $\forall v \in C_i$, v has at least two neighbours in C_i. T_i is the rest of the domain. T_i can be viewed as a forest of trees with roots in C_i. (These roots are not in T_i). We denote these graphs after the termination of the game by D, C and T.

The following lemma is crucial:

Lemma 1. *If $C_i \neq \emptyset$, then there are no hanging trees of depth greater than* 2.

Proof. By contradiction. Consider the first round (say r) in which there is a hanging tree $T^{(3)}$ of depth 3. That means that in the round $r-1$ there was a hanging tree $T^{(2)}$ of depth 2 and from one of its leaves at depth 2 (say from a vertex v) a new link was explored. Let the root of $T^{(2)}$ be a vertex $t \in C_{r-1}$. It has two neighbours in C_{r-1}. Take the embedding $\mathcal{E}$ used by the adversary at round $r-1$ and modify it locally to embed $T^{(2)}$ such as in Figure 12. This modification is indeed possible, because the place for the vertex v is either free or is occupied by another branch of $T^{(2)}$ (which can be exchanged with the branch leading to v). If this place belongs to C_{r-1} or to another tree, then in the previous round the adversary should have directed the growth of $T^{(2)}$ to this place, thus constructing smaller domain.

The resulting embedding $\mathcal{E}'$ (together with the choice of explored link (v, w)) results in smaller domain compared with $\mathcal{E}$, which is the contradiction with respect to our choice of the adversary.

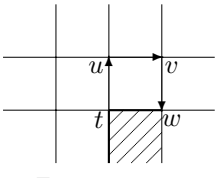

Figure 12.

It is easy to see that $C_i = \emptyset$ holds only at the very beginning of the computation. Using similar arguments as in the proof above we can show that once there is a tree of depth 4, the adversary will turn its branches to form a cycle.

So far we have shown that the adversary can limit the depth of hanging trees by looping their branches back to the core graph. Another way to reduce the depth of hanging trees is to connect two trees, thus eliminating trees that grow deep. We will refine our adversary by making it prefer the following option:

– If there are several possibilities (for embedment and choice of explored links) resulting to the domains of the same size, prefer the option in which a tree branches loop back to the core graph rather than the option in which trees come in touch. (See Figure 13)

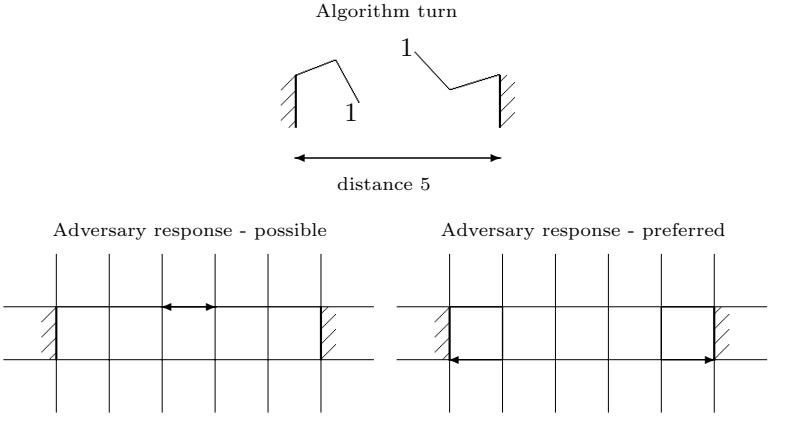

Figure 13.

Lemma 1 allows us to examine how the core graph grows:

Lemma 2. *If $C_i \neq \emptyset$, then there exists an ear decomposition of $C_{i+1} - C_i$ such that each ear has the length at most* 4.

More precisely, there exists a sequence of graphs $C_i = C_{i,0}, C_{i,1}, \ldots, C_{i,k} = C_{i+1}$ such that for each j, $0 \leq j < k$, $C_{i,j+1} - C_{i,j}$ is a path of length at most 4 *which starts and ends in $C_{i,j}$.*

Proof. First note that there is an ear decomposition with ears of length at most 6. See that new vertices are added to the core graph only when branches of some tree loop back to the core graph or when two trees meet. (The third possibility occurs when two branches of the same tree meet. But this case is not possible under our adversary, because the adversary preferes to loop branches back to the core graph.) In the first case an ear of length at most 3 is formed (since trees are of depth at most 2, plus newly explored link), in the second case each tree can contribute by a path of length at most 3, bounding the overall length of the new created ear by 6.

To form a *long ear* of length 5 or 6, at least one tree must contribute by a path of length 3. Our adversary prefers looping back branches of trees to the core graph. It may happen that adversary cannot loop back branch of length 3, because that will increase the size of the domain. One such situation is shown on Figure 14.

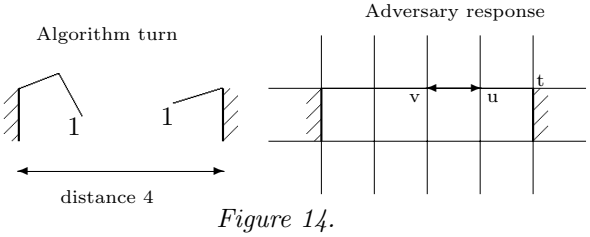

Figure 14.

However, that is possible because the tree of the vertex u contributed by only 2 links $((t,u),(u,v))$ and there were overlapping links from different trees $((u,v)$

and (v, u)). If there are not overlapping links (there is no request for exploration from u), looping back to the core graph is possible, because it will not enlarge the domain. Similarly, looping back is possible if both trees contribute by paths of length 3, as shown in Figure 13.

That means that if there is still a long ear, then it must also have small loop(s) at its end(s). We can first add an ear formed by this (these) small loop(s), decreasing the overall length of the long ear to at most 4. (It is easy to see that an ear of length 6 must have small loops at both of its ends.)

Let E_C be the number of links in the core graph C. We will prove our lower bound by proving the lower bound on the expression

$$\frac{E_C + |T|}{|C| + |T|} = \frac{\text{total number of explored links}}{N} \tag{1}$$

Let C_0 be the initial core graph (formed when the first loop is closed) and let E_{C_0} be the number of links in it. Let C_i be the core graph after adding i ears according to Lemma 2 and let E_{C_i} be the number of links in it. Let T_i be the set of hanging trees at that moment. Following Lemma 1, we can bound $|T_i|$ by the number of vertices at the distance at most 2 from C_i.

We are interested in the ratio

$$\frac{E_{C_k} + |T_k|}{|C_k| + |T_k|} = \frac{E_{C_0} + |T_0| + \sum_{i=1}^{k}(E_{C_i} - E_{C_{i-1}} + |T_i| - |T_{i-1}|)}{|C_0| + |T_0| + \sum_{i=1}^{k}(|C_i| - |C_{i-1}| + |T_i| - |T_{i-1}|)} \tag{2}$$

Denote $E_{C_i} - E_{C_{i-1}}$ by e_i, it represents the number of links in the i-th ear. Similarly $|C_i| - |C_{i-1}| = v_i$, the number of inner vertices in the i-th ear. Clearly $e_i = v_i + 1$. Let t_i be the number of vertices that are at distance at most 2 from C_i, but they are at greater distance from C_{i-1}. $|T_i| - |T_{i-1}|$ can be estimated as $t_i - v_i$, since v_i inner vertices of the i-th ear are transferred from T_{i-1} to C_i. (Note that all inner vertices of the i-th ear are inside T_{i-1}, because the ear has length at most 4.)

We can rewrite (2):

$$\frac{E_{C_0} + |T_0| + \sum_{i=1}^{k}(e_i + (t_i - v_i))}{|C_0| + |T_0| + \sum_{i=1}^{k}(v_i + (t_i - v_i))} = \frac{E_{C_0} + |T_0| + \sum_{i=1}^{k}(t_i + 1)}{|C_0| + |T_0| + \sum_{i=1}^{k} t_i} \tag{3}$$

Let $t = max_{1 \le i \le k} t_i$. Because $\sum_{i=1}^{k}(t_i + 1) / \sum_{i=1}^{k} t_i \ge (t + 1)/t$, we get

$$\frac{E_{C_k} + |T_k|}{|C_k| + |T_k|} \ge \frac{E_{C_0} + |T_0| + k(t + 1)}{|C_0| + |T_0| + kt} \tag{4}$$

Since $|C_0|$, E_{C_0}, $|T_0|$ and t are in $O(1)$, we can write $|C_0| + |T_0| + kt = k't$ and $E_{C_0} + |T_0| + k(t+1) = k'(t+1) - O(1)$ for some $k' > k$. Applying to (4) we get

$$E_{C_k} + |T_k| \ge \frac{k'(t + 1) - O(1)}{k't}(|C_k| + |T_k|) = \frac{t + 1}{t} \cdot (|C_k| + |T_k|) - O(1) \tag{5}$$

since $k't \geq |C_k| + |T_k|$.

Note that this expression is in more general form than the simple lower bound for ceased broadcasting. It says that any algorithm that has reached r vertices must have used $r(t+1)/t - O(1)$ links.

All we need now is to bound t:

Lemma 3. *t can be bounded by 7 for ears of length at most 4.*

Proof. First note that there is only a finite (and not really high) number of possible cases which can be tested by computer. We perform the case analysis.

We will use the fact that each vertex of C has at least two neighbours in C.

Consider an ear of 4 links and 3 vertices u, v and w being added to C. These three vertices either lie on a line or not. In the first case, all possible situations (up to the symmetry) are shown in Figure 15.

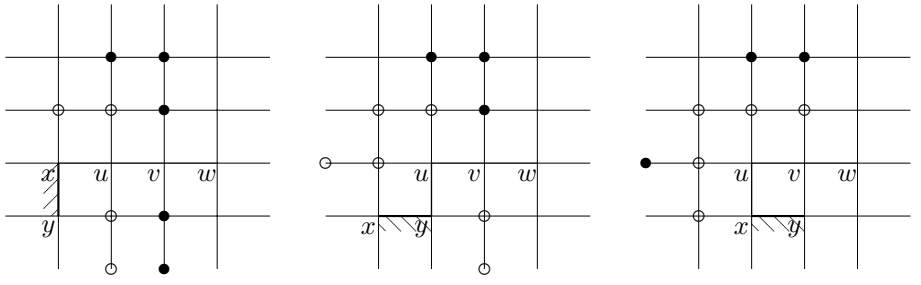

Figure 15.

x and y are vertices in C. Empty circles represent vertices in 2–neighbourhood of C. Full circles represent vertices potentially added to the 2–neighbourhood of C by adding the ear uvw.

Due to the symmetry only the left part from the vertical axis passing through v is shown. t is in these cases bounded by 6, 4 and 5, respectively. (If the right part is mirror image of the left part. Otherwise t is even smaller.)

If u, v and w don't lie on a line, then four possible cases are shown in Figure 16. Again it is sufficient to consider only one half (left bottom) of the situation.

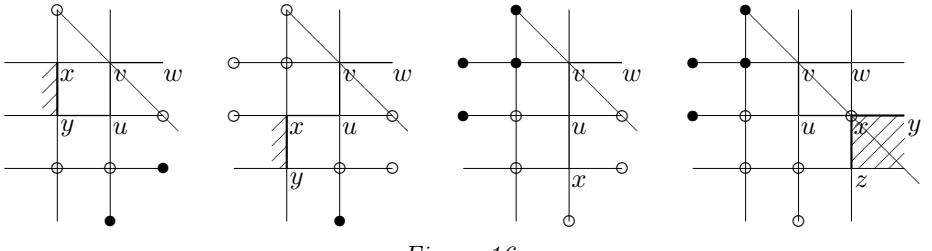

Figure 16.

t is in these cases bounded by 4, 2, 7 and 7, respectively.

Ears of smaller length are handled similarly. All possible cases can be drawn on previous figures, just with smaller number of t – vertices.

Now we can apply (5):

Proposition 1. *Any broadcasting algorithm on unoriented tori that reached r vertices must have used at least $8/7r - O(1)$ links in the worst case.*

Corollary 1. *Broadcasting on unoriented N–node tori requires the use of at least $8/7N - O(1)$ links, even in synchronous case.*

5 Conclusions

We have presented improved upper and lower bounds for broadcasting on an unoriented torus. The main question is how to narrow or close the gap between these bounds.

We believe that the lower bound can be improved. A possible way of improvement can be obtained by the analysis of the following situation. The highest t ($t = 7$) is reached by adding ears that are not sustainable. Including such an ear prevents from further inclusion of an ear with $t = 7$ on involved vertices. Ears with smaller t must be added to prepare the ground for another ear with $t = 7$. Further improvement can be based on the following hypothesis: *There exists an adversary such that in each completed computation there are no hanging trees of depth 2 or more (although during the computation there could be some).*

We note that the algorithm from [DKP96] can be modified to compute relative addresses (w.r.t. the starting vertex) of all vertices in the torus. Our algorithm cannot be modified in such a way. It would be interesting to prove (or disprove) the lower bound of $2N - o(N)$ messages for this problem.

References

DKP96. Diks, K. – Kranakis, E. – Pelc, A.: *Broadcasting in Unlabeled Tori.* Département d'Informatique, Université du Québec á Hull, Technical Report RR 96/12-5, 1996.

DR97. Dobrev, S. - Ružička, P.: *Linear Broadcasting and N log log N Election in Unoriented Hypercubes.* Proc. of the 4th International Colloquium on Structural Information and Communication Complexity (SIROCCO'97), Carleton Press, Ascona, Switzerland, July 1997, pp. 55–73.

DRT98. Dobrev, S. - Ružička, P. - Tel, G.: *Time and Bit Optimal Broadcasting on Anonymous Unoriented Hypercubes.* Proc. of the 5th International Colloquium on Structural Information and Communication Complexity (SIROCCO'98), Carleton Press, Amalfi, Italy, June 1998.

DDKPR98. Diks, K. - Dobrev, S. - Kranakis, E. - Pelc, A. - Ružička, P.: *Broadcasting in Unlabeled Hypercubes with Linear Number of Messages.* To appear in Information Processing Letters, 1998.

FMS95. Flocchini, P. – Mans, B. – Santoro, N.: *Sense of Direction: Formal Definitions and Properties.* Proc. of the 1st International Colloquium on Structural Information and Communication Complexity (SIROCCO'94), Carleton Press, 1995, pp. 9–34.

FMS96. Flocchini, P. – Mans, B. – Santoro, N.: *On the Impact of Sense of Direction on Communication Complexity.* Information Processing Letters, Vol. 63(1), July 1997, pp. 23–31.

Mans96. Mans, B.: *Broadcast, Traversal and Election in Unlabelled Hypercube.* Proc. of the 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO'96), Carleton Press, Siena, Italy, June 1996, pp. 333–334.

Mans96b. Mans, B.: *Optimal Distributed Algorithms in Unlabeled Tori and Chordal Rings.* Proc. of the 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO'96), Carleton Press, Siena, Italy, June 1996, pp. 17–31.

Pel97. Peleg, D.: Personal communication at the Sienna Research School'97 on "Compact Routing and Sense of Direction", Siena, Italy, June 1997.

Pet85. Peterson, G. L.: *Efficient algorithms for elections in meshes and complete networks.* Technical Report TR140, Dept. of Computer Science, Univ. of Rochester, Rochester, NY 14627, 1985.

Tel94a. Tel, G.: *Network Orientation.* International Journal of Foundations of Computer Science 5, 1994, pp. 23–57.

Tel94b. Tel, G.: *Introduction to Distributed Algorithms.* Cambridge University Press, Cambridge, 1994.

Tel95a. Tel, G.: *Sense of Direction in Processor Networks.* In: SOFSEM'95, Theory and Practise of Informatics, LNCS 1012, Springer–Verlag, 1995, pp. 50–82.