Broadcasting in unlabeled tori

Krzysztof Diks [*] Evangelos Kranakis [†] Andrzej Pelc [‡]

Abstract

We consider broadcasting a message from one node to all other nodes of an asynchronous totally unlabeled torus: neither nodes nor links have *a priori* assigned labels but they know the topology and the size of the torus. Nodes can send messages of arbitrary size and we are interested in minimizing the total number of messages. A naive broadcasting algorithm in a $n \times n$ totally unlabeled torus uses $3n^2 + 1$ messages, while the obvious lower bound is $n^2 - 1$. The main result of this paper is a broadcasting algorithm using $2n^2 + O(n)$ messages. We also give a lower bound of $1.04n^2 - O(n)$ messages. This is the first result on message complexity of broadcasting in totally unlabeled networks.

[*]**Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02-097 Warszawa, Poland.**
E-mail: diks@mimuw.edu.pl
This work was done during the author's visit at the Université du Québec à Hull.

[†]**School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada.**
E-mail: kranakis@scs.carleton.ca
Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) grant.

[‡]**Département d'Informatique, Université du Québec à Hull, Hull, Québec J8X 3X7, Canada.**
E-mail: Andrzej_Pelc@uqah.uquebec.ca
Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) grant.

1 Introduction

The study of the computational power of anonymous networks is an important and well established research area in distributed computing (cf. [12] and the survey [7]). In an anonymous network processors do not have distinct identities and execute identical algorithms. The impossibility of distinguishing processors yields symmetry in computations and restricts the power of the network both in terms of the class of functions that can be computed and the time and cost of computations. The study of anonymous networks was initiated in [1] and then pursued by many authors, both for specific networks, such as rings [2, 3, 10] and hypercubes [8] and for arbitrary networks [4, 9, 13]. Among the problems studied in this context are the following: Which functions a given anonymous network can compute? What is the message and bit complexity of such computation? Is it possible to perform leader election in the network?

An even more severe restriction on the amount of knowledge available to processors is the lack of *sense of direction* (cf. [11, 5]), when there is no "consistent" labeling of links. Networks in which neither links nor nodes have *a priori* assigned labels are obviously both anonymous and devoid of sense of direction. We will call such networks *totally unlabeled*.

In this paper we consider *broadcasting* in one of the most popular and often implemented topologies: the *torus*. (See, e.g., the survey [6] for references to the literature on broadcasting.) One node, called the *source*, has a message which has to be transmitted to all other nodes. We assume that the torus is totally unlabeled. This implies that a node sending a message does not know which neighbor will receive it. The only knowledge available to a node is the topology of the torus, its size, and an extra bit signifying "I am the source" (in case of the source) or "I am not the source" (in case of all remaining nodes). All other knowledge has to be acquired in the communication process. Nodes can send messages of arbitrary size and content, thus conveying already gained knowledge together with the source message. Only nodes that already got the source message can transmit. The network is asynchronous: every message sent on a link arrives to its destination in finite but unbounded time. Links are of FIFO type: messages sent on a given link arrive in the order of sending. All processors are identical and run the same algorithm.

We are interested in minimizing the number of messages needed to achieve broadcasting. In the naive algorithm every node relays the message to all neighbors except those from which it got it. Due to asynchronicity this may take as many as $2l - (k-1)$ messages in a network with k nodes and l links, i.e., $3n^2 + 1$ messages for the $n \times n$ torus. On the other hand, the minimal number of messages is $k-1$, even in a labeled network, i.e., $n^2 - 1$ for the torus. The main result of this paper is a broadcasting algorithm using $2n^2 + O(n)$ messages. We also give a lower bound of $1.04n^2 - O(n)$ messages. This is the first result on message complexity of broadcasting in totally unlabeled networks.

The paper is organized as follows. In section 2 we fix the terminology and give a precise definition of the model and of a broadcasting algorithm in a totally unlabeled torus. Section 3 is devoted to the construction and analysis of such an efficient broadcasting algorithm. In section 4 we prove the above mentioned lower bound on message complexity of broadcasting. Section 5 contains conclusions and open problems.

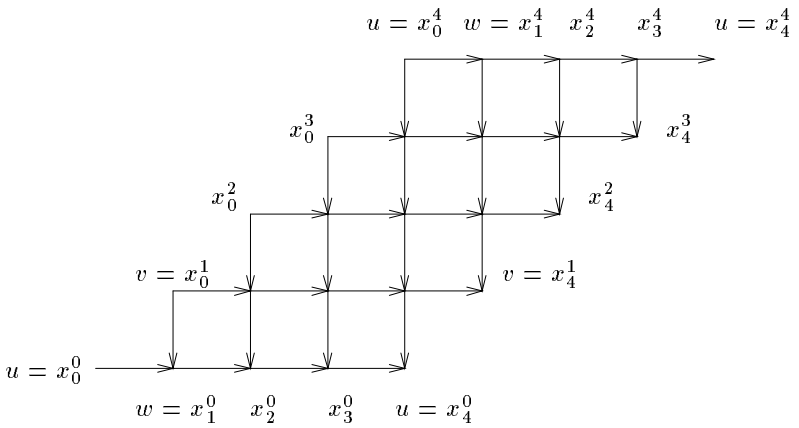

Figure 1: Orientation $\mathcal{O}(u, v, w)$ and the canonical labeling corresponding to the generator (u, v, w).

2 The model

We first fix the terminology and state basic facts concerning graph theoretic aspects of the network. The *torus* T_n, for $n > 2$, is the graph $G = (V, E)$ in which $V = \{(i, j) : 0 \leq i, j < n\}$ and neighbors of every node (i, j) are $(i + 1, j)$, $(i - 1, j)$, $(i, j + 1)$, and $(i, j - 1)$. (All arithmetic operations on coordinates of pairs (i, j) are done modulo n.) For every node (i, j) its *diagonal neighbors* are nodes $(i + 1, j + 1)$, $(i + 1, j - 1)$, $(i - 1, j + 1)$, and $(i - 1, j - 1)$. Notice that diagonal neighbors are not neighbors in the graph. A generator in T_n is any triple (u, v, w) of nodes such that v is a diagonal neighbor of u and w is one of the two common neighbors of u and v.

Proposition 2.1 *For any generator (u, v, w) there exists exactly one authomorphism ψ of the torus T_n, such that $\psi(u) = (0, 0)$, $\psi(v) = (1, 1)$, and $\psi(w) = (1, 0)$. This authomorphism ψ is said to be* generated *by the generator (u, v, w).*

Let ψ be the authomorphism generated by a fixed generator (u, v, w). The ith *diagonal* of T_n, for $i = 0, ..., n$ is the sequence of nodes $D_i = (x_i^0, x_i^1, ..., x_i^n)$, such that $x_i^j = \psi^{-1}(i + j, j)$. In particular, $x_0^0 = x_0^n = u$, $x_0^1 = v$, and $D_0 = D_n$. For $j = 0, ..., n$, the jth *row* of T_n is the sequence of nodes $R_j = (x_0^j, x_1^j, ..., x_n^j)$. In particular, $x_0^0 = x_n^0 = u$, $x_1^0 = w$, and $R_0 = R_n$. For a given generator (u, v, w), the above defined labeling x_i^j of nodes of T_n will be called *canonical*. The label of a node in the canonical labeling will be called its *identity*.

Consider the orientation $\mathcal{O}(u, v, w)$ of edges of T_n in which every edge is oriented from a node in D_i to a node in D_{i+1}, for $i = 0, ..., n - 1$ (cf. Figure 1).

Proposition 2.2 *For any generator (u, v, w) every node has two incoming and two outgoing edges in orientation $\mathcal{O}(u, v, w)$.*

We now proceed to the definition of the communication model for the unlabeled torus. For any node v, $I(v)$ denotes the set of edges incident to v. All links from $I(v)$ are connected to v

by distinct *ports*. Each port has two buffers: one for storing incoming and the other for storing outgoing messages. Before the start of broadcasting each node v of T_n knows n and knows if it is the source. There is no *a priori* labeling of links or nodes, i.e., v does not know which link from $I(v)$ joins it with which neighbor. For every v we fix an arbitrary local labeling of ports of v. For convenience, node v identifies its ports with respective links, i.e., this labeling is a bijection $\sigma_v : \{1,2,3,4\} \to I(v)$. It should be stressed that those labelings are chosen arbitrarily by all nodes and no "coherence" among them is assumed.

Messages placed in the outgoing buffer of a given port of v are sent to the neighbor joined with v by the respective link. Links are of FIFO type, i.e., messages transmitted through a given link arrive to the destination in the same order as they were placed in the buffer. The network is asynchronous: every message sent on a link arrives to its destination in finite but unbounded time. Whenever a message arrives at a node, it appears in the incoming buffer of the respective port and is received. We use two primitive operations:

$send_v(M,i)$ denotes the operation of sending message M by node v through port $\sigma_v(i)$.

$receive_v$ denotes the function which waits for the first incoming message at v and returns the couple (M,i), where M is the message and i is the port number at which this message came.

Let s denote the source of broadcasting, and $\Sigma = \{\sigma_v : v \in V\}$ - the set of fixed local labelings of ports. We want to formalize the assumption that nodes can send arbitrary messages and the decision of a node at any point of broadcasting can be based on the entire knowledge that it acquired so far. This current knowledge of a node v increases with each incoming message: it is the previous knowledge of v plus the newly obtained knowledge. What knowledge can be gained by getting a new message? The following three items can be learned:

- the current knowledge of the sender in the moment of sending;
- the local port number on which the sender sent the message;
- the local port number on which the message was received.

We define a *history* in T_n to be any finite sequence of pairs $[a_1,b_1]$,..., $[a_j,b_j]$, where $j > 0$ and $a_i, b_i \in \{\text{source}, \text{dest}, 1, 2, 3, 4\}$, for all $i \leq j$. Let $\mathcal{H}$ be the set of all histories in T_n. A *chooser* is any function

$$\phi : \mathcal{H} \to P(\{1,2,3,4\}) \times \{stop, wait\}.$$

The role of the chooser is the following: for any node v and any current history (i.e., knowledge) available at this node, it chooses the (local labels of) ports on which a message (containing everything v knows) will be sent. It also selects one of the states *stop* or *wait*. The first one indicates that all actions of v should be terminated after sending messages on the chosen ports - no further messages will be read by the processor even if they arrive at some ports in the future; the second state means that the node is still active and it will read the next incoming message.

For any node v the *initial history* of v is defined as follows: $IH_v = [source, source]$ if $v = s$ and $IH_v = [dest, dest]$ if $v \neq s$. Thus the initial history at node v specifies only whether v is or is not the source.

Fix a chooser ϕ. For every node v, $\mathcal{A}_\phi(v)$ denotes the following algorithm executed at v:

begin
$CH_v := IH_v$; //initialization of current history//
repeat
$(X, Y) := \phi(CH_v)$;
for all $i \in X$ **do**
$send_v([CH_v, i], i)$; //the message containing the current history of v and the port number i on which it is sent, is sent on port i, for all i chosen by the chooser//
if $Y = wait$ **then**
begin
$([H, out], in) := receive_v$;//a message is received on port in//
$CH_v := CH_v \bullet H \bullet [out, in]$; //current history is updated by concatenating to its previous value the history obtained in the new message and the information about the local port numbers on which it was sent and received//
end
until $Y = stop$
end.

A *protocol* for T_n with chooser ϕ is the family $\mathcal{A}_\phi = \{\mathcal{A}_\phi(v) : v \in V\}$. A protocol $\mathcal{A}_\phi$ is a *broadcasting algorithm* from source s iff the following conditions are satisfied:

1. $\phi(IH_v) = (X, Y)$, where $\emptyset \neq X \subset \{1, 2, 3, 4\}$ and $s = v$, or $X = \emptyset$, $Y = wait$ and $v \neq s$ (in the beginning the source has to send a message to some neighbor, while all other nodes have to wait and cannot send anything (before obtaining the first message);

2. $\phi(CH_v) = (X, Y)$, where $X \subset \{1, 2, 3, 4\}$ (for any current history at any node the chooser chooses a subset of available ports to send messages);

3. for any set of local labelings Σ every algorithm $\mathcal{A}_\phi(v)$ assignes value *stop* to the variable Y.

Condition 3 is crucial in the above definition. In the beginning $Y = wait$ in all nodes except possibly the source. Hence the requirement that value *stop* be assigned means that each node has received a message and each node has finished its work. The requirement that this should happen for all labelings captures the full anonymity of broadcasting: eventually all nodes get the message, although at each transmission the choice to which neighbor a node sends a message is controled by the adversary. In an actual implementation of a broadcasting algorithm the original source message is appended to each "history" message specified above.

The *message complexity* of a broadcasting algorithm $\mathcal{A}_\phi$ for a given set Σ of local labelings, is the number C_Σ of operations *send* performed during the execution of the algorithm. The *message complexity* of $\mathcal{A}_\phi$ is the maximum of numbers C_Σ, over all sets Σ of local labelings.

The description of the algorithm in section 3 will be informal: for reasons of clarity of presentation we will not define the chooser formally. Indeed, in most cases, a node does not need to know the entire history to decide on which ports it should send messages. Likewise, it usually

does not need to send its entire history but only some crucial facts that can be deduced from it. Here is an example of such a fact: "I got a message from the source via a path of length 4 from one neighbor and via a path of length 8 from another neighbor". Nodes can also send conditional orders to neighbors. Formally such orders are encoded in the chooser, e.g. "if you got a message previously then stop, otherwise send a message to one neighbor and tell it to stop". In particular, each node knows from its history on which ports it already got or sent messages. Call all other ports of a node *free*. Thus the instruction "send a message on all your free ports" is legitimate and will be often used. We will use the phrase "a node stops after getting a message" if the value of the chooser is $(\emptyset, stop)$, i.e., when a node does not perform any further actions after getting this message. This way of describing anonymous algorithms better conveys their main idea and it will be clear in each case how to transform such an informal description into chooser definition.

3 The broadcasting algorithm

In this section we describe a broadcasting algorithm in the totally unlabeled torus T_n and compute its message complexity. The algorithm works in three stages. The aim of the first stage is the identification of a generator (s, v, w). In the second stage we identify the diagonal D_0 corresponding to this generator. More precisely, all nodes of this diagonal get the source message and each of them computes its identity with respect to the generator (s, v, w). Moreover, each node in D_0 identifies ports of both of its outgoing edges with respect to the orientation $\mathcal{O}(s, v, w)$. Finally, in the third stage, the remaining diagonals D_1, ..., D_{n-1} are identified and thus all remaining nodes get the source message.

We now describe each of the stages in a detailed way. Fix any set Σ of local labelings.

Stage I.

Let u_1, u_2, u_3 and u_4 be neighbors of s corresponding to ports at s labeled 1, 2, 3, and 4, respectively. Denote by $u_{i,j}$ the common neighbor of u_i and u_j, different from s. Denote by u'_i the neighbor of u_i which is different from s and is not a diagonal neighbor of s (cf. Figure 2).

The source s sends messages on all ports. After getting the message, each neighbor u_i of s sends a message on all its free ports. All neighbors of these four nodes, except s (i.e., nodes $u_{i,j}$ and u'_i), wait for the second message after getting the first one. (Notice that only nodes $u_{i,j}$ can get two messages at this point.) After getting two messages (from u_i and u_j) node $u_{i,j}$ confirms the receipt of two messages by sending a message to both senders. This ends the actions of $u_{i,j}$ in Stage I. After getting two such confirmations, each node u_i sends a message on the port corresponding to u'_i (now this port is identified) containing an order of ending Stage I. After getting this order, each node u'_i confirms its receipt and ends its actions in Stage I. Additionally, node u_1 sends the message "you are my neighbor $u_{1,j}$ with smaller j" to the respective neighbor. After getting the confirmation from u'_i node u_i reports the termination of Stage I to the source and ends its actions in Stage I. After getting all four termination reports, the source s ends Stage I.

Upon completion of Stage I we fix the generator (s, v, w), where $w = u_1$ and v is the node $u_{1,j}$ with smaller j. All nodes of this generator know their identities (in the corresponding canonical labeling): $x_0^0 = s$, $x_0^1 = u_{1,j}$, $x_1^0 = u_1$. Moreover, node u'_1 knows its identity x_2^0. The outgoing edges for s (in orientation $\mathcal{O}(s, v, w)$) are edges corresponding to ports 1 and k, where $u_{1,k}$ is the diagonal

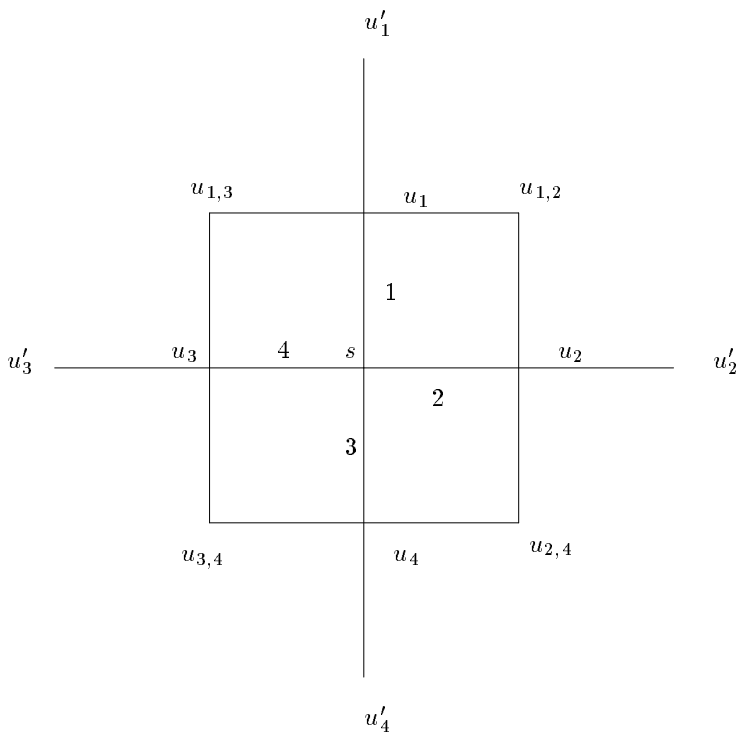

Figure 2: Stage I.

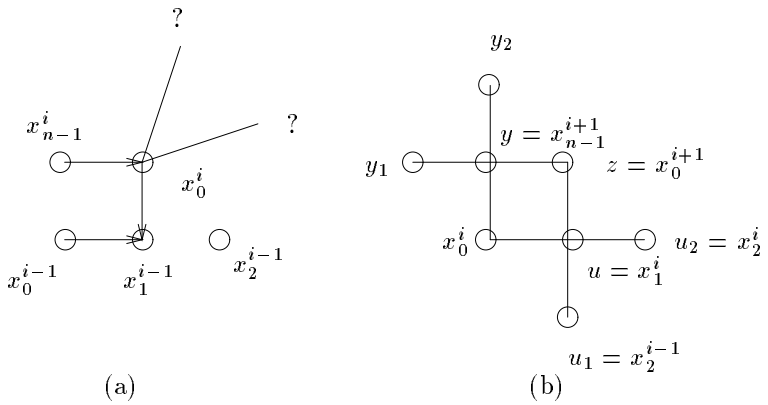

Figure 3: Step i of stage II.

neighbor of s, other than v. The other two edges incident to s are incoming. s knows that port 1 corresponds to $x^0_1 = u_1$. Node x^0_1 knows which of its ports corresponds to x^1_0 and x^1_0 knows which of its ports corresponds to x^0_1. The edge joining x^1_0 and x^0_1 is outgoing for x^1_0. Node x^1_0 also knows which port corresponds to x^1_{n-1}; the corresponding edge is incoming for x^1_0. (Recall that $x^1_0 = x^1_n$).

Stage II.

Stage II works in $n-1$ steps. There are two aims for step $i = 1, ..., n-1$: x^i_0 should classify all its incident edges into incoming and outgoing, and nodes x^{i+1}_0, x^i_1 and x^i_2 should learn their identities. We will maintain the following invariants in the beginning of each step $i = 1, ..., n-1$ (cf. Figure 3 (a)):

- Node x^{i-1}_0 knows its identity, knows (port labels of) its incoming and outgoing edges and knows which port corresponds to x^{i-1}_1.
- Node x^{i-1}_1 knows its identity and knows which of its ports corresponds to x^i_0.
- Node x^i_0 knows its identity and knows which of its ports corresponds to x^{i-1}_1. The edge joining x^i_0 and x^{i-1}_1 is outgoing for x^i_0. Node x^i_0 also knows which of its ports corresponds to x^i_{n-1}; the corresponding edge is incoming for x^i_0 (Recall that $x^i_0 = x^i_n$).
- Node x^{i-1}_2 knows its identity.

Notice that upon completion of Stage I, in the beginning of step 1 of Stage II, all these invariants are maintained. We now describe step i of Stage II.

Node x^{i-1}_0 sends a message "start step i" on the port corresponding to x^{i-1}_1 and ends its actions in step i. Node x^{i-1}_1 relays this message to x^i_0 and ends its actions in step i.

Let u and y denote neighbors of x^i_0 corresponding to edges not yet classified by x^i_0 (recall that, by the invariant, x^i_0 has already classified two of its incident edges). We have $\{u, y\} = \{x^i_1, x^{i+1}_{n-1}\}$. W.l.o.g. assume that $u = x^i_1$. Let z be the common neighbor of u and y, different from x^i_0. Clearly, $z = x^{i+1}_0$. Denote by u_1, u_2 and y_1, y_2 the neighbors of u and y, respectively, different from x^i_0 and x^{i+1}_0. W.l.o.g. assume that $u_1 = x^{i-1}_2$ (cf. Figure 3 (b)).

After getting the "start step i" message, node x_0^i sends messages on ports corresponding to edges not yet classified (i.e., to neighbors u and y). Each of these nodes sends a message on all ports except the one corresponding to x_0^i (on which the message arrived), i.e., u sends a message to u_1, u_2 and z, and y sends a message to y_1, y_2 and z. After getting the message, node $u_1 = x_2^{i-1}$, which has already learned its identity before step i, can compute the identity of the sender, as it expects only one message in this step. It sends to the sender the information "your identity is x_1^i" and ends its actions in step i.

Node u, after learning its own identity x_1^i, sends this information on ports corresponding to u_2 and z. Node z, after getting three messages (two from u and one from y) knows its identity x_0^{i+1} and knows the identity x_1^i of u. It also knows the identity x_{n-1}^{i+1} of y. Hence it deduces that the edge joining it with y is incoming and the edge joining it with u is outgoing. Node z sends this information (call the corresponding message m) to neighbors u and y and ends its actions in step i.

Node y, after obtaining message m from z, sends the message "terminate step i" to neighbors y_1 and y_2. Nodes y_1 and y_2 confirm the receipt of this message and end their actions in step i. After getting both confirmations, node y sends the message "terminate step i" to x_0^i and ends its actions in step i.

Node u, after obtaining message m from z, sends a message "your identity is x_2^i" to u_2 (at this point u has already identified the respective port and computed the identity of u_2). Node u_2 confirms the receipt of this message and ends its actions in step i. After getting this confirmation, node u sends the message "terminate step i" to x_0^i and ends its actions in step i. After getting message "terminate step i" from both y and u, node x_0^i ends its actions in step i. This concludes step i. All invariants are maintained.

If $i < n-1$, node x_0^i initiates step $i+1$; if $i = n-1$, Stage II is terminated. Upon its completion each node from the diagonal D_0 knows its identity and its outgoing edges.

Stage III.

Every node $x \in D_0$ sends a message on ports corresponding to both of its outgoing edges. Then it stops. Every other node x waits to get two messages in Stage III. These messages can come only on incoming edges. Then x learns to which diagonal D_i it belongs (messages came from nodes on diagonal D_{i-1}), and computes its own identity. If $i < n-1$, node x sends a message on ports corresponding to both of its outgoing edges. Then it stops. If $i = n-1$, node x stops.

This concludes the description of our algorithm. The following is our main result.

Theorem 3.1 *There exists a broadcasting algorithm in the totally unlabeled torus T_n, using $2n^2 + O(n)$ messages.*

Proof: We compute the message complexity of the above described algorithm. Stage I uses a constant number of messages (more precisely, 37 messages). Stage II uses $O(n)$ messages, as each step uses a constant number of messages (more precisely, a total of $23(n-1)$ messages are used). Stage III uses $2n(n-1)$ messages (every node in diagonals D_0, ..., D_{n-2} sends two messages). □

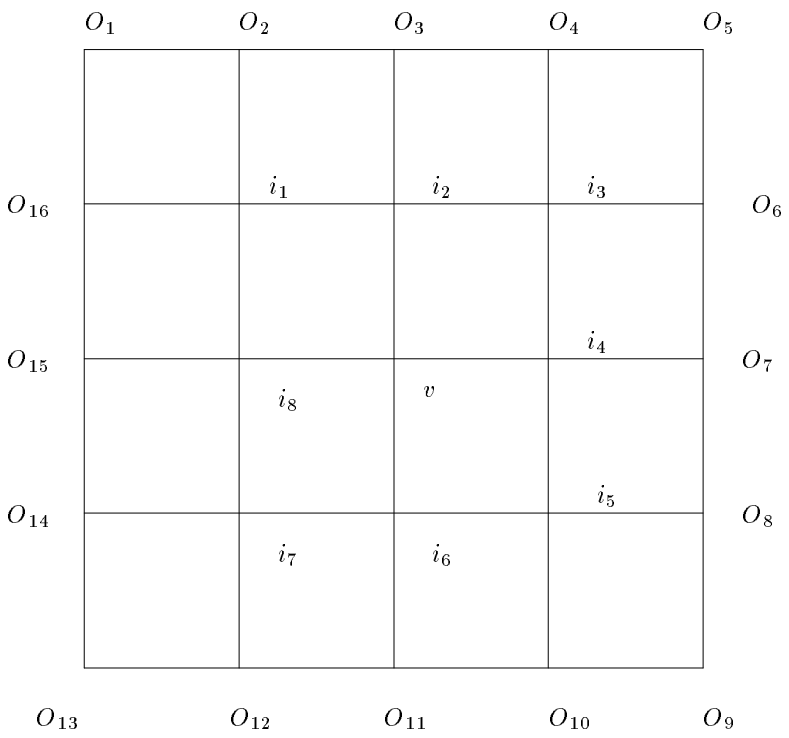

Figure 4: The subgraph $C(v)$.

4 The lower bound

The trivial lower bound on message complexity of broadcasting in T_n is $n^2 - 1$: this is the message complexity of broadcasting in a labeled torus. Under the totally unlabeled scenario we increase this lower bound by a small multiplicative constant.

Theorem 4.1 *The message complexity of any broadcasting algorithm in a totally unlabeled torus T_n is at least* $1.04n^2 - O(n)$.

Proof: We will prove that if $\mathcal{A}_\phi$ is any broadcasting algorithm in T_n then there exists a set of local labelings Σ for which $\mathcal{A}_\phi$ uses at least $(n^2 - 1) + \lfloor \frac{n-1}{5} \rfloor^2$ messages. W.l.o.g. assume that $n > 5$. For any node v, let $C(v)$ be the subgraph of T_n shown in Figure 4. Assume that the source s is outside of $C(v)$. Nodes O_j are called *outer* nodes and all other nodes (i_j and v) are called *inner* nodes of $C(v)$. Node v is called the *center* of $C(v)$.

Claim. For any broadcasting algorithm $\mathcal{B}$ in T_n and any set $\Sigma = \{\sigma_u : u \in V\}$ of local labelings, there exists a set $\Sigma' = \{\sigma'_u : u \in V\}$ of local labelings, such that

- $\sigma'_u = \sigma_u$ for u outside of $C(v)$;
- some node in $C(v)$ gets at least two messages in the execution of $\mathcal{B}$ under Σ'.

Proof of the Claim: Suppose not. Fix a set $\Sigma = \{\sigma_u : u \in V\}$ of local labelings and fix a node v as above. Let $F(v)$ be the subgraph of $C(v)$ spanned by those edges on which messages are transmitted. Orient each edge of $F(v)$ in the direction of transmission. By our assumption, $F(v)$ is a forest of rooted trees, where roots are these nodes of $C(v)$ which have in-degree 0 in $F(v)$. We will use expressions "child" and "ancestor" with respect to the forest $F(v)$. Denote by S_u the maximal subtree of $F(v)$ with root u. Let w be the closest ancestor of v which is an outer node in $C(v)$ and let d be the length of the path P in $F(v)$ from w to v. Consider three cases.

Case 1. $d \geq 4$.

Let w_0, w_1, w_2, w_3, w_4 be the first 5 nodes on the path P. W.l.o.g. assume that $w_0 = O_2$. Hence $w_1 = i_1$. Define new labelings σ'_x, $x \in \{i_1, i_2, O_3\}$ as follows:

$$\sigma'_{i_1}(\{i_1, i_2\}) = \sigma_{w_1}(\{w_1, w_2\}),\ \sigma'_{i_2}(\{i_2, i_1\}) = \sigma_{w_2}(\{w_2, w_1\}),$$

$$\sigma'_{i_2}(\{i_2, O_3\}) = \sigma_{w_2}(\{w_2, w_3\}),\ \sigma'_{O_3}(\{O_3, i_2\}) = \sigma_{w_3}(\{w_3, w_2\}),$$

$$\sigma'_{O_3}(\{O_3, O_2\}) = \sigma_{w_3}(\{w_3, w_4\}).$$

Let $\Sigma' = \Sigma \setminus \{\sigma_{i_1}, \sigma_{i_2}, \sigma_{O_3}\} \cup \{\sigma'_{i_1}, \sigma'_{i_2}, \sigma'_{O_3}\}$. Under the set Σ' of local labelings the node $w_0 = O_2$ will get two messages, contradiction.

Case 2. $d = 3$.

Let the path P be w_0, w_1, w_2, w_3 with $w = w_0 = O_2$ and $w_1 = i_1$. If the height of tree S_w is larger than 3 then there exists a path of length 4 in S_w starting at w, and we can argue as in case 1. Hence suppose that this height is 3.

Consider node O_3. If it is not in S_{O_2} or is a child of O_2 in S_{O_2} then it gets the message via a path outside of $C(v)$ or, respectively, directly from O_2. On the other hand, it is possible to change local labelings at nodes i_1 and i_2, so that O_3 gets this message as before and another message via the path O_2, i_1, i_2. Contradiction. Thus O_3 must have depth 3 in S_{O_2}, i.e., it must be a leaf in this tree.

Consider node O_1. If it were a child of O_2 in S_{O_2} it would get two messages under a modified set of local labelings, similarly as O_3. Hence i_1 must be the only child of O_2 in S_{O_2}.

On the other hand, O_{15} must belong to S_{O_2}, for otherwise it would get two messages under a modified set of local labelings, similarly as O_3. It follows that there are two edge-disjoint paths of length 2 in S_{i_1}: one going from i_1 to O_3 via i_2, and the other going from i_1 to O_{15} via O_{16} or i_8. Now it is possible to change local labelings at i_1, i_2 and i_8, so that under the modified set of local labelings node v gets two messages, one from i_2 and the other from i_8. Contradiction.

Case 2. $d = 2$.

Let the path P be w_0, w_1, w_2. W.l.o.g. assume that $w = w_0 = O_3$ and hence $w_1 = i_2$. As before, we may assume that the height of S_w is at most 3. Nodes O_2 and O_4 must be children of O_3 in S_{O_3}, for otherwise the local labeling at O_3 could be changed to prevent v from getting the message. Similarly, i_1 and i_3 must be children of i_2 in S_{i_2}. It follows that O_2 must be a leaf in S_{O_3}, for otherwise the local labeling at O_2 could be changed so that i_1 would get two messages. Thus O_1 must get the message from a neighbor x different from O_2. Now, local labelings at O_3 and O_2 can be changed, so that under the modified set of local labelings node O_1 gets two messages, one as in the old labeling (from neighbor x) and the other one via path O_3, O_2, O_1. Contradiction.

This concludes the proof of the Claim. Now it is easy to finish the proof of the theorem. There are at least $k = \lfloor \frac{n-1}{5} \rfloor^2$ pairwise dijoint graphs $C(v)$ not containing the source. Let $v_1, ..., v_k$ be their centers. We will show that for any broadcasting algorithm $\mathcal{A}_\phi$ there exists a set of local labelings Σ for which at least one node of every graph $C(v_l)$ gets at least two messages. This will prove our result.

Let $\mathcal{A}_\phi$ be a broadcasting algorithm and Σ_0 any set of local labelings. If Σ_0 has the above property, we are done. Otherwise let $i \leq k$ be the least index for which all nodes in $C(v_i)$ get exactly one message. By the Claim, it is possible to transform Σ_0 to Σ_1, by changing local labelings only for nodes in $C(v_i)$, so that some node of $C(v_i)$ gets at least two messages under Σ_1. If Σ_1 has the desired property, we are done. Otherwise, let $j \leq k$ be the least index for which all nodes in $C(v_j)$ get exactly one message under Σ_1 . Clearly $j > i$. Next we modify Σ_1 to Σ_2 by changing local labelings only for nodes in $C(v_j)$, so that some node of $C(v_j)$ gets at least two messages under Σ_2. Since modifications are local to $C(v_j)$, the property that some node of $C(v_i)$ gets at least two messages holds under Σ_2 as well. After at most k modifications we get a set of local labelings Σ for which at least one node of *every* graph $C(v_l)$ gets at least two messages. □

5 Conclusion

We presented upper and lower bounds on message complexity of broadcasting in a $n \times n$ torus. The natural open problem is to narrow or even close the gap between those bounds. It also seems interesting to investigate broadcasting in other unlabeled networks and see how message complexity changes with respect to the labeled scenario. In a forthcoming paper we study k-node networks in which broadcasting under the totally unlabeled scenario can be done using $k - 1$ messages, i.e., as efficiently as in the labeled networks. We also propose efficient broadcasting in the totally unlabeled hypercube.

References

[1] D. Angluin, Local and Global Properties in Networks of Processors, Proc. 12th Annual ACM Symposium on Theory of Computing (1980), 82-93.

[2] H. Attiya, M. Snir and M. Warmuth, Computing on an Anonymous Ring, Journal of the ACM, 35 (1988), 845-875.

[3] H. Attiya and M. Snir, "Better Computing on the Anonymous Ring, Journal of Algorithms 12 (1991), 204-238.

[4] P. Boldi, S. Shammah, S. Vigna, P. Gemmell, B. Codenotti, and J. Simon, Symmetry Breaking in Anonymous Networks: Characterizations, Proc. 4th Israel Symposium on Theory of Computing and Systems (1996), 16-26.

[5] P. Flocchini, B. Mans and N. Santoro, "Sense of Direction: Formal Definitions and Properties", Proc. 1st Int. Conf. on Structural Information and Communication Complexity (1994), 9-34, Carleton University Press, 1995.

[6] S.M. Hedetniemi, S.T. Hedetniemi and A.L. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1988), 319-349.

[7] E. Kranakis, Symmetry and Computability in Anonymous Networks: A Brief Survey, Proc. 3rd Int. Conf. on Structural Information and Communication Complexity (SIROCCO'96), to appear, Carleton University Press, 1997.

[8] E. Kranakis and D. Krizanc, Distributed Computing on Anonymous Hypercube Networks, Journal of Algorithms, to appear.

[9] E. Kranakis, D. Krizanc and J. van der Berg, Computing Boolean Functions on Anonymous Networks, Information and Computation, 114 (1994), 214-236.

[10] S. Moran and M. K. Warmuth, Gap Theorems for Distributed Computation, Proc. 5th ACM Symposium on Principles of Distributed Computing (1986), 131-140.

[11] N. Santoro, Sense of Direction, Topological Awareness, and Communication Complexity, ACM SIGACT News 16 (1984), 50 - 56.

[12] G. Tel, Introduction to Distributed Algorithms, Cambridge University Press, 1996.

[13] M. Yamashita and T. Kameda, Computing on an Anonymous Network, Proc. 7th Annual ACM Symposium on Principles of Distributed Computing (1988), 117 - 130.