Broadcasting in Unlabeled Hypercubes with Linear Number of Messages

Krzysztof Diks * Stefan Dobrev § Evangelos Kranakis † Andrzej Pelc ‡
Peter Ružička §

Abstract

We consider broadcasting a message from one node to all other nodes of an asynchronous totally unlabeled n-node hypercube. Neither nodes nor links have *a priori* assigned labels. We show a broadcasting algorithm which uses only $O(n)$ messages. This answers affirmatively an open question of G. Tel [17].

Keywords: Broadcasting, Unlabeled Hypercube.

1 Introduction

The study of the computational power of anonymous networks is an important and well established research area in distributed computing (cf. [16] and the survey [9]). 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, 14] and hypercubes [10] and for arbitrary networks [4, 11, 18].

*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.
§Institute of Informatics, Faculty of Mathematics and Physics, Comenius University, Mlynská dolina 842 15 Bratislava, Slovak Republic.
E-mail: ruzicka@alpha.dcs.fmph.uniba.sk
Partially supported by EU Grant No. INCO-COP 96-0195 "ALTEC-KIT" and by Slovak VEGA project 1/4315/97.
†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.

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. [15, 7]), when there is no "consistent" labeling of links. For rings, the sense of direction is equivalent to orientation and its computational aspects were studied, e.g., in [2]. *Totally unlabeled* networks, those in which neither links nor nodes have *a priori* assigned labels, are obviously both anonymous and devoid of sense of direction.

One of the fundamental tasks in distributed computing is *broadcasting*. (See, e.g., the survey [8] for references to the literature on broadcasting.) One node of the network, called the *source*, has a message which has to be transmitted to all other nodes.

In this paper (cf. [5, 6]) we are concerned with broadcasting in totally unlabeled hypercubes. This implies that a node sending a message to its neighbor knows neither its own identity nor that of the neighbor. Moreover, a node sending a message on a yet unused port does not know which neighbor will receive it. The only knowledge *a priori* available to a node is the topology of the hypercube, its own degree (from which the size of the hypercube can be inferred) 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.

The lack of sense of direction during the broadcasting process can be modeled in the following way. In the beginning, every node fixes an arbitrary *local* labeling of its ports. It should be stressed that these labelings are not guaranteed to be consistent in any way. At any stage of broadcasting, a node knows which messages it has previously sent or received on which ports but it has no knowledge concerning yet unused ports. Thus, at any stage, an informed node can make three decisions, based on information locally available to it at this time. Either not send any message, or send a message on a *specific* already used port, or decide to send a message on a yet unused port. In the latter case, the actual choice of the port on which the message will be sent is made by the adversary, as all these ports look alike to the sending node and we are interested in the worst case.

Messages are of size $O(\log n)$, where n is the number of nodes, and of arbitrary content: They may convey 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 message complexity of broadcasting, i.e., the number of messages used in the worst case. The worst case refers to adversary decisions concerning choices of yet unused ports on which informed nodes send messages. In the naive broadcasting algorithm working in a totally unlabeled network, every node relays the message to all neighbors except those from which it got it. This yields an algorithm with message complexity linear in the number of links, thus $\Theta(n \log n)$ for an n-node hypercube. Tel [17] (cf. also Mans [13]) asked if there exists a broadcasting algorithm in the totally unlabeled n-node hypercube, using $O(n)$ messages in the worst case. We answer this question in the affirmative and show such a broadcasting algorithm.

The following communication task, related to broadcasting, is called *traversal* (cf. [16]). A token starting at one node of a network has to visit all other nodes, moving along links. To every move of the token a control message can be attached. The difference between broadcasting and traversal is that in the latter the token cannot be multiplied on its way. (In other words, nodes need not only learn the source message but have to see the original of a document.) Another question in [17, 13] was whether there exists a traversal algorithm in the totally unlabeled n-node hypercube, using $O(n)$ token moves in the worst case. A small modification of our broadcasting algorithm gives a positive answer to this question as well.

The paper is organized as follows. In section 2 we give graph theoretic preliminaries concerning the hypercube. In section 3 we present an outline of our algorithm. In section 4 we give its detailed description, prove its correctness and estimate its message complexity.

2 Preliminaries

For any set B and positive integer k, B^k denotes the set of all sequences of length k whose terms are in B. If $x \in B^k$, we denote by $x_{i,j}$ the segment of x from position i to position j. We denote by a^k the only element of the set $\{a\}^k$. For two sequences x and y from $\{0,1\}^k$, their *Hamming distance* is the number of positions on which they differ.

Let $B = \{0,1\}$. For any positive integer k, the *hypercube of dimension* k is the graph $H_k = (V_k, E_k)$ in which $V_k = B^k$ and $E_k = \{\{u,v\} : u \text{ and } v \text{ are at Hamming distance 1 }\}$. For $Y \subset B^k$, we denote by $H_k(Y)$ the subgraph of H_k induced by Y. Let Z be a proper subset of $\{1,...,k\}$. We denote by $B^k(Z)$ the subset of B^k consisting of all those sequences that have 0 on positions from Z.

Proposition 2.1

1. *Every pair of distinct nodes of H_k have either 0 or 2 common neighbors.*
2. *Nodes u and v have 2 common neighbors iff their Hamming distance is 2.*
3. *If $u = (u_1,...,u_k)$ and $v = (v_1,...,v_k)$ differ on positions i and j then their common neighbors are $x = (u_1,...,\overline{u_i},...,u_k)$ and $y = (u_1,...,\overline{u_j},...,u_k)$, where $\overline{u_l} = 1 - u_l$.*

Let $l \leq k$ be positive integers. Let u be a node in H_k and u_1, ..., u_l some of its neighbors. Define the following sequence of sets:

$$S_0 = \{u\};$$

$$S_1 = \{u_1,...,u_l\};$$

$$S_i = \{w \in B^k \setminus S_{i-2} : w \text{ has 2 distinct neighbors in } S_{i-1}\},$$

for $i = 2,...,l$. Define $S(u,u_1,...,u_l) = \bigcup_{i=0}^{l} S_i$. The sets S_i will be called *layers* of $S(u,u_1,...,u_l)$.

Proposition 2.2 *$H_k(S(u, u_1, ..., u_l))$ is isomorphic with H_l.*

Proposition 2.3 *There exists an automorphism ψ of the hypercube H_k, such that $\psi(u) = 0^k$, $\psi(u_1) = 10^{k-1}$,..., $\psi(u_l) = 0^{l-1}10^{k-l}$.*

The automorphism ψ is said to be *consistent* with $(u, u_1, ..., u_l)$.

Proposition 2.4 *For all automorphisms ψ_1 and ψ_2 consistent with $(u, u_1, ..., u_l)$ and any node $v \in S(u, u_1, ..., u_l)$, $\psi_1(v) = \psi_2(v)$.*

Let w be any node from $S(u, u_1, ..., u_l)$ and let $w_1, ..., w_{k-l}$ be neighbors of w outside of $S(u, u_1, ..., u_l)$. The set $S(w, w_1, ..., w_{k-l})$ is said to be *orthogonal* to $S(u, u_1, ..., u_l)$ at node w.

A *dominating set* in a graph G is a set D of nodes such that every node of G either belongs to D or is adjacent to a node from D.

Proposition 2.5 *Let D be a dominating set in $H_k(S(u, u_1, ..., u_l))$ and v any node of H_k. There exists a node $w \in D$ such that either v is a node of $H_k(S(w, w_1, ..., w_{k-l}))$ or is adjacent to a node of this graph, where $S(w, w_1, ..., w_{k-l})$ is orthogonal to $S(u, u_1, ..., u_l)$.*

A *1-error-correcting code* of length k is a subset C of B^k such that any pair of sequences from C are at Hamming distance at least 3.

Proposition 2.6 *(Hamming, cf. [12]) If $k = 2^r - 1$, for some positive integer r, then there exists a 1-error-correcting code of length k and size 2^{k-r}. It is called a perfect Hamming code.*

Proposition 2.7 *If $k = 2^r - 1$ then there exists a dominating set of size 2^{k-r} in H_k.*

Proof: Take a perfect Hamming code $C \subset B^k$. For any $c \in C$ let $K(c)$ be the set of elements of B^k at Hamming distance at most 1 from c (the neighborhood of c in H_k). By the property of C the sets $K(c)$ are pairwise disjoint and each of them has size $k + 1 = 2^r$. Their union has size $2^r \cdot 2^{k-r} = 2^k$ and consequently covers all nodes. Thus C is a dominating set. $\square$

Proposition 2.8 *The hypercube H_k has a dominating set of size at most $2^{k+1}/k$.*

Proof: Let k' be the largest integer of the form $2^r - 1$, not exceeding k. Clearly $k' \geq \frac{k}{2}$. Represent H_k as a product $H_{k'} \times H_{k-k'}$ and consider the $2^{k-k'}$ copies of $H_{k'}$. In each of them there exists a dominating set of size $2^{k'-r}$. The union of these sets over all copies is a dominating set in H_k of size $2^{k-k'} \cdot 2^{k'-r} = 2^{k-r} \leq 2^{k+1}/k$. $\square$

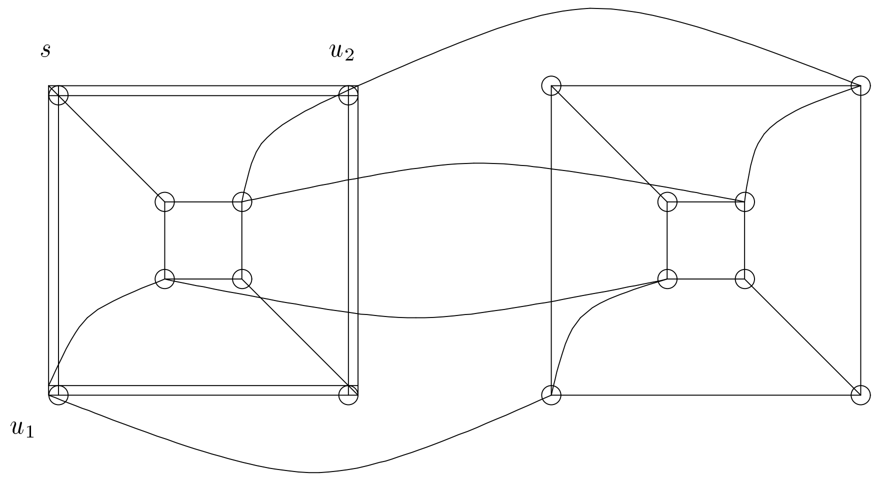

Figure 1: Broadcasting in hypercube H_4 from source s. Double lines represent edges of $S(s, u_1, u_2)$ and curved lines edges of orthogonal subcube. u_1, u_2 are nodes of the dominating set in $S(s, u_1, u_2)$.

3 Outline of the algorithm

We now describe a broadcasting algorithm in the hypercube H_k, that uses $O(2^k)$ messages. Let $n = 2^k$. Suppose that $k > 3$ and let $l = \lceil k/2 \rceil$. Fix a dominating set D in H_l of size at most $2^{l+1}/l$ (cf. Proposition 2.8) and a set of local labelings.

We first give an overview of the algorithm (cf. Figure 1). It works in two stages. The start and the end of each stage is decided by the source s. The aim of the first stage is the identification of nodes of the graph $H_k(S(s, u_1, ..., u_l))$, where u_1, ..., u_l are neighbors of s, such that i is the port number at s corresponding to the edge joining u_i with s. These nodes are identified by assigning to each $v \in S(s, u_1, ..., u_l)$ the label $\psi(v)$, where ψ is an automorphism of H_k consistent with $(s, u_1, ..., u_l)$ (cf. Propositions 2.3, 2.4). We say that a node $v \in S(s, u_1, ..., u_l)$ is a *chief* iff $\psi(v)_{1,l} \in D$, i.e.,if v is an element of a dominating set in $H_k(S(s, u_1, ..., u_l))$ isomorphic with D. Upon completion of the first stage each node $v \in S(s, u_1, ..., u_l)$ will know from its identifier if it is a chief or not.

Let v be a chief and let $1 \le i_1 < ... < i_m \le k$ be port numbers at v corresponding to edges joining v with nodes outside of $S(s, u_1, ..., u_l)$. Let v_1, ..., v_m be these corresponding adjacent nodes. The aim of the second stage is identifying, for every chief v, all nodes of the set $S(v, v_1, ..., v_m)$ orthogonal to $S(s, u_1, ..., u_l)$ and transmitting the source message to all nodes of $S(v, v_1, ..., v_m)$ and all their neighbors. Since chiefs form a dominating set in $H_k(S(s, u_1, ..., u_l))$, all nodes of H_k will get the message, by Proposition 2.5.

4 Detailed description of the algorithm and its correctness

Stage I.

This stage consists of $l+1$ steps $0, ..., l$. The start and the termination of each step is decided by the source s. Simultaneously we will construct BFS trees $T_0, ..., T_l$ rooted at s. The tree T_i will be a spanning tree of the set $S_0 \cup ... \cup S_i$, where S_i are layers of $S(s, u_1, ..., u_l)$. The tree T_i will be used to communicate between the source and nodes in layers S_0, ..., S_i. It will be maintained in the following distributed way:

- every node $v \in S_j$, $j = 1, .., i$, keeps the least port number $p(v)$ corresponding to an edge joining it with a node from S_{j-1};
- every node $v \in S_j$, $j = 0, .., i-1$, keeps the set $C(v)$ of ports corresponding to edges $e = \{v, u\}$, such that $u \in S_{j+1}$ and e corresponds to the port $p(u)$ at node u.

In the description of the algorithm we will show how these variables are updated. For brevity we will identify numbers $p(v)$ and $l \in C(v)$ with neighbors of v to which these ports correspond. Node $p(v)$ will be called *parent* of v and nodes $l \in C(v)$ will be called *children* of v.

Step 0.

No messages are sent. The source s assigns to itself the identifier $\psi(s) = 0^k$ and initializes the set $C(s)$ as empty. The tree T_0 consists only of the root s.

Step 1.

The source s sends messages to nodes u_1,...,u_l (i.e., on ports 1,...,l) with the order "confirm". After l confirmations it updates $C(s) := \{1, ..., l\}$ and terminates the step. Every node u_i learns the port number i on which s sent the message, assigns to itself the identifier $\psi(u_i) = 0^{i-1}10^{k-i}$, sets $p(u_i)$ to the port number on which it got the message, initializes the set $C(u_i)$ as empty and sends confirmation to s (on port $p(u_i)$).

Step i, $i = 2, ..., l$.

s starts the step by sending a "start step i" message to all nodes of layer S_{i-1}. The message travels via edges of the tree T_{i-1}. Every node $v \in S_{i-1}$ knows that it is in this layer from previous messages and already has an identifier. After getting the message that step i has started, v sends its identifier $\psi(v)$ on all free ports, i.e., to neighbors outside of S_{i-2}. Node v will have exactly $l-i+1$ neighbors in S_i. It waits for $l-i+1$ confirmations that neighbors in S_i have been informed, identified and have terminated step i.

The confirmation process works as follows. A node u that got a message from a node in S_{i-1} waits until i such messages arrive from i distinct nodes in S_{i-1}. When this happens, u decides that it belongs to layer S_i and assigns to itself the identifier $\psi(u)$ calculated from any two identifiers $\psi(v_1)$ and $\psi(v_2)$ obtained from nodes in S_{i-1}. This is done unambiguously by Proposition 2.1. Then u initializes $C(u)$ as the empty set, sets $p(u)$ to the least port number on which it got a message from

S_{i-1} and sends to all its neighbors in S_{i-1} a message saying that it terminated step i. Moreover, it includes the information "you are my parent" in the message to $p(u)$. All nodes u that got less than i messages from layer S_{i-1} (in fact just one message), wait for further instructions.

After getting $l-i+1$ confirmations from (distinct) neighbors in layer S_i, node v from S_{i-1} updates the set $C(v)$ to the set of ports on which the message "you are my parent" arrived. Then it sends an "exclusion" message on all ports on which no confirmation arrived, except the ports leading to nodes from S_{i-2}. (Node v knows which are these ports because it got messages on these ports in step $i-1$.) The "exclusion" message says "you are not in $S(s, u_1, ..., u_l)$". A node x which got this message sends back a confirmation to the sender and terminates its work in step i. After getting all expected confirmations from these neighbors outside of $S(s, u_1, ..., u_l)$, node v sends a message that it terminated work in step i to its parent. Every node y in T_i whose set of children $C(y)$ is nonempty waits for the message from all its children that they terminated work in step i. Then it sends such a termination message to its parent.

Step i ends when the source s gets the termination message from all its children. Stage I ends with the end of step l. Upon completion of Stage I all nodes in $S(s, u_1, ..., u_l)$ got the source message and have an identifier. In particular, every node in this set knows whether it is a chief.

Stage II.

The source s starts Stage II by sending a "start Stage II" message to all nodes of $S(s, u_1, ..., u_l)$ via edges of tree T_l. After getting this message, every chief v starts the construction of the set $S(v, v_1, ..., v_m)$ orthogonal to $S(s, u_1, ..., u_l)$, where v_1, ..., v_m are its neighbors outside of $S(s, u_1, ..., u_l)$. The construction procedure is almost identical to that in Stage I. The only difference is that the identifier of chief v has to be carried throughout the construction and a new node w decides that it belongs to the ith layer of $S(v, v_1, ..., v_m)$ after getting i messages from nodes of the $(i-1)$th layer of *the same set* $S(v, v_1, ..., v_m)$. This is an important precaution: some nodes from $S(s, u_1, ..., u_l)$ may be adjacent to many chiefs and consequently a node w can get messages from layers in different sets (constructed by different chiefs). It should not confuse these messages. In the process of constructing the set $S(v, v_1, ..., v_m)$ all neighbors of elements of this set get the source message (cf. Stage I).

After constructing the set $S(v, v_1, ..., v_m)$, the chief v is "done". All non-chiefs in $S(s, u_1, ..., u_l)$ are "done" right after getting the "start Stage II" message. A leaf in T_l which is "done" sends a "final termination" message to its parent in T_l and stops. Every other node except the source sends a "final termination" message to its parent (and stops) after getting such a message from all its children and after being "done". The source stops under the same conditions. This concludes Stage II and the entire algorithm.

Theorem 4.1 *There exists a broadcasting algorithm of message complexity $O(n)$ in the totally unlabeled n-node hypercube.*

Proof: We will compute the message complexity of the above described algorithm.

Stage I.
In step 0 no messages are sent. In step 1 the source sends l messages and obtains l confirmations. In every step $1 < i \leq l$ every node u from layer S_{i-1} sends $k-i+1$ messages, gets $l-i+1$ confirmations, then sends $k-l$ messages and gets $k-l$ confirmations. Additionally, a message is sent twice on all edges of the tree T_{i-1}. Thus the total number of messages in Stage I is at most

$$2l + \sum_{i=2}^{l}(2(|S_1| + ... + |S_{i-1}|) + ((k-i+1) + (l-i+1) + 2(k-l)) \cdot |S_{i-1}|) \leq$$

$$2l + \sum_{i=2}^{l} 2 \cdot 2^l + \sum_{i=2}^{l} 4k \cdot |S_{i-1}| \leq 2l + 2l \cdot 2^l + 4k \cdot 2^l = \Theta(\sqrt{n}\log n),$$

where $n = 2^k$.

Stage II.
Sending the "start Stage II" and "final termination" messages via edges of tree T_l uses $2 \cdot (2^l - 1) = \Theta(\sqrt{n})$ messages. The argument used to find message complexity of Stage I shows that the construction of each set $S(v, v_1, ..., v_m)$ orthogonal to $S(s, u_1, ..., u_l)$, for a chief v, uses $\Theta(\sqrt{n}\log n)$ messages. By Proposition 2.8 there are $\Theta(\frac{\sqrt{n}}{\log n})$ chiefs in $S(s, u_1, ..., u_l)$, hence all these constructions use $\Theta(n)$ messages. It follows that the message complexity of the algorithm is $\Theta(n)$. □

5 Conclusion

In this paper we have given the first broadcasting algorithm that uses a linear number of messages on a totally unlabeled n-node hypercube, with each message of size $O(\log n)$. Thus the total number of bits transmitted during the execution of our algorithm is $O(n \log n)$. This raises the interesting open question of whether or not an $\Omega(n \log n)$ lower bound is possible for this problem on a 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] K. Diks, E. Kranakis and A. Pelc, Broadcasting in Unlabeled Networks, Technical Report RR 96/12-5, Université du Québec à Hull, 1996.

[6] S. Dobrev and P. Ružička, Linear Broadcasting and N log log N Election in Unoriented Hypercubes, Proc. 4th Coll. on Structure, Information and Communication Complexity (SIROCCO'97), to appear.

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

[8] S.M. Hedetniemi, S.T. Hedetniemi and A.L. Liestman, A survey of Gossiping and Broadcasting in Communication Networks, Networks 18 (1988), 319-349.

[9] E. Kranakis, Symmetry and Computability in Anonymous Networks: A Brief Survey, Proc. 3rd Int. Conf. on Structural Information and Communication Complexity (SIROCCO'96), Carleton University Press, 1997, 1-16.

[10] E. Kranakis and D. Krizanc, Distributed Computing on Anonymous Hypercube Networks, Journal of Algorithms 23 (1997), 32-50.

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

[12] F.J. MacWilliams and N.J.A. Sloane, The Theory of Error-Correcting Codes, Elsevier Science Publ. B.V. 1977.

[13] B. Mans, Broadcast, Traversal and Election in Unlabelled Hypercube, Proc. 3rd Coll. on Structure, Information and Communication Complexity (SIROCCO'96), Carleton University Press, 1997, 333-334.

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

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

[16] G. Tel, Introduction to Distributed Algorithms, Cambridge University Press, 1994.

[17] G. Tel, Sense of Direction in Processor Networks, Proc. SOFSEM'95, Theory and Practice of Informatics, LNCS 1012, Springer-Verlag, 1995, 50-82.

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