A Tradeoff Between Information and Communication in Broadcast Protocols

(Revised version)

Baruch Awerbuch [1] *Oded Goldreich* [2] *David Peleg* [3] *Ronen Vainish* [2]

Abstract

This paper concerns the message complexity of broadcast in arbitrary point-to-point communication networks. *Broadcast* is a task initiated by a *single* processor that wishes to convey a message to all processors in the network. We assume the widely accepted model of communication networks, in which each processor initially knows the identity of its neighbors, but does not know the entire network topology. Although it seems obvious that the number of messages required for broadcast in this model equals the number of links, no proof of this basic fact has been given before.

We show that the message complexity of broadcast depends on the exact complexity measure. If messages of unbounded length are counted at unit cost, then broadcast requires $\Theta(|V|)$ messages, where V is the set of processors in the network. We prove that if one counts messages of *bounded length* then broadcast requires $\Theta(|E|)$ messages, where E is the set of edges in the network.

Assuming an intermediate model in which each vertex knows the topology of the network in radius $\rho \geq 1$ from itself, we prove matching upper and lower bounds of $\Theta(\min\{|E|, |V|^{1+\frac{\Theta(1)}{\rho}}\})$ on the number of messages of bounded length required for broadcast. Both the upper and the lower bounds hold for both synchronous and asynchronous network models.

The same results hold for the construction of spanning trees, and various other global tasks.

[1]Dept. of Math., MIT, Cambridge, MA 02139, baruch@theory.lcs.mit.edu Partially supported by Air Force contract TNDGAFOSR-86-0078, ARO contract DAAL03-86-K-0171 and NSF contract CCR8611442.

[2]Dept. of Computer Science, Technion, Haifa, 32000, Israel. Partially supported by the New York Metropolitan Research Fund.

[3]Dept. of Computer Science, Stanford University, Stanford CA 94305. Partially supported by ONR contract N00014-85-C-0731.

1 Introduction

Broadcast [DM] is one of the most fundamental tasks in distributed computing. It is initiated by a single processor, called the *source*, wishing to distribute a message (*the initial message*) to all processors in the network.

We consider the standard model of distributed computing, which is a point-to-point communication network. The network is modeled by an undirected graph $G(V, E)$ whose vertices represent processors and whose edges represent bidirectional communication links (cf. [A1, Bu, FL, GHS]). Communication itself is either synchronous or asynchronous – all our results hold for both cases. An *elementary* message may contain only a constant number of bits and a constant number of vertex identities. Longer messages must be chopped into elementary messages prior to transmission. The *communication complexity* of an algorithm is the total number of (elementary) messages sent in a worst-case execution.

The two most basic and well-known algorithms for broadcast in a point-to-point communication network are *tree broadcast* and *flooding*. The tree broadcast algorithm requires the existence of a spanning tree that is known to all processors. Given such a tree, broadcast can be performed with only $|V| - 1$ messages. In case such a tree is not available, it has to be constructed first. Note, however, that the problem of constructing a spanning tree from a single initiator is equivalent in terms of communication complexity to the problem of broadcasting a single message. This follows from the fact that any broadcast algorithm can also be used to build a tree in the network; the parent of a node in that tree is the neighbor from which the first message is received.

In contrast, the flooding algorithm makes no initial assumptions. This algorithm achieves its task by simply forwarding the message over *all* links. Clearly, this requires $\Theta(|E|)$ messages.

When discussing the applicability of these (and other) broadcast algorithms to a communication network, a central issue is the amount of knowledge available at the vertices regarding the topology of the network. There are two common models, representing the two possible extreme situations. In the first model (which we denote KT_∞ for reasons which will become clear later) one assumes that every vertex has full knowledge of the network topology. In this model, it is obvious that broadcast can be performed with the minimal number of messages, i.e., the communication complexity of the problem is $\Theta(|V|)$. This is because each vertex can use its knowledge in order to locally construct the (same) spanning tree without sending any message. Then, the tree broadcast algorithm can be applied.

The standard model for a communication network, which we denote KT_1, assumes very little knowledge. That is, initially each processor knows only its own identity and the identity of its neighbors, but nothing else. In this model, a well-known "folk theorem" asserts that flooding is the best that can be done, i.e., that $\Theta(|E|)$ is a tight bound for the communication complexity of the problem. However, to the best of our knowledge, no proof of this lower bound (or for that matter, of any lower bound higher than $\Omega(|V|)$) was given before. At first glance the claim seems obvious. Indeed, the claim *is* obvious if we consider the even more extreme "anonymous" model KT_0, based the (unnatural) assumption that a vertex does not know even the identities of its neighbors. The intuition behind the $\Omega(|E|)$ lower bound for KT_0 is that in this case "every edge must be traversed at least once". However, slightly shifting from this extreme model towards the more common (and more reasonable) KT_1 model, this intuition fails, as is implied by the following algorithm.

Consider a "traveler" which performs a Depth-First Search (DFS) traversal (cf. [E]) on the communication graph. Observe that by carrying the list of vertices visited so far, the traveler may avoid traversing non-tree edges (or "backward" edges) since at any point during the search the traveler knows which vertices have already been visited. Thus, the traveler will not traverse every graph edge, but only $n-1$ tree edges (each being traversed exactly twice).

While this algorithm indicates that there is no need to traverse each graph edge, it does not disprove the above "folk theorem". Indeed, observe that the total number of *elementary* messages sent is not $2|V|$, but rather $O(|V|^2)$, as the lists carried by the traveler may contain up to $O(|V|)$ vertex identities; thus the traversal of an edge may require $O(|V|)$ elementary messages.

In this paper, we (finally) prove the above "folk theorem" for the standard KT_1 model. More precisely, we show that in a communication network where each vertex knows only its neighbors, the number of elementary messages required for broadcast is $\Omega(|E|)$.

Theorem 1: *For every graph $G(V,E)$ there exists a related family C_G containing $|E|$ graphs of $2|V|$ vertices and $2|E|$ edges each, such that any protocol that works correctly on all graphs of C_G sends $\Omega(|E|)$ elementary messages over a constant fraction of the graphs of C_G. This lower bound holds even if the network is synchronous, all the vertices start the protocol at the same round, and the vertices know the size of the network.*

Once we establish this gap between the two extreme models, it becomes interesting to look at intermediate points, in which processors are allowed only partial knowledge of the topology, and investigate the implications of such knowledge with regard to the communication complexity of the broadcast operation. These intermediate points attempt to capture common situations in which vertices know more about their near-by vicinity than about other regions of the network. We formalize such situations by introducing a (mainly theoretical) hierarchy of models KT_ρ (for every integer $\rho \geq 0$) in which, loosely speaking, every vertex knows the topology of a subgraph of radius ρ around it. Hence the models KT_0 and KT_1 described earlier correspond to the lowest two levels of this hierarchy, while KT_∞ corresponds to the highest levels, i.e., the models KT_ρ with ρ being the diameter of the network or larger.

For this hierarchy of models, we prove a general tradeoff result. For every fixed $\rho \geq 1$, the number of elementary messages required for broadcast in the model KT_ρ is $\Theta(\min\{|E|, |V|^{1+\frac{\Theta(1)}{\rho}}\})$. To be more precise, we can prove the following.

Theorem 2: *There exists a constant $c' > 0$ such that for every two integers $\rho \geq 1$ and $n \geq 1$ there exists a family F_ρ of graphs with m edges and n vertices each, where $m = \Omega(n^{1+\frac{c'}{\rho}})$, such that any protocol that works correctly on all graphs of F_ρ in the model KT_ρ sends at least $\Omega(m/\rho)$ messages over a constant fraction of the graphs of F_ρ. This lower bound holds even if the network is synchronous, all the vertices start the protocol at the same round, and the size of the network is known to each vertex.*

Theorem 3: *There exists a constant $c > 0$ such that for every integer $\rho \geq 1$ and for any graph $G(V, E)$, broadcast can be performed in the model KT_ρ using at most $O(\min\{|E|, |V|^{1+\frac{c}{\rho}}\})$ messages. This upper bound holds even if the network is asynchronous.*

Our results suggest that there exists an inherent tradeoff between the information that the vertices have about the communication graph, and the number of messages needed to perform the broadcast. The more knowledgeable vertices are about the network, the cheaper it is to perform broadcast.

One should not confuse our problem of constructing a tree from a single initiator with the harder problem of constructing a tree when the algorithm is initiated by (possibly) multiple vertices (or the strongly related *leader-election* problem). The latter problem is itself a very basic problem in distributed computing, since it is equivalent to a variety of other problems (e.g., counting, computing majority or parity, finding a leader, etc.).

Most previously known lower bounds on the leader election problem, as well as lower bounds on various related problems, were proved in the KT_∞ model, i.e., for networks whose topology is *known* to all vertices, and in particular, networks with a very *regular* structure. Among others, network topologies considered in lower bounds proofs include: rings [AAHK, ASW, Bu, F, FL, GS, MW, MZ, PKR], cliques [AG, F, KMZ1, KMZ2], toruses [GI], meshes [F], binary trees [F] and others. Other results, e.g. [KMZ1], are obtained in the other extreme model, KT_0, and strongly rely on the assumption that processors do not a-priori know the identities of their neighbors.

One of the novelties of our work is that it applies to a network of arbitrary topology, and takes *full* advantage of the fact that network's topology is initially *unknown*. For example, as a corollary we get also that constructing a spanning tree in a network whose topology is *unknown* is harder than constructing a spanning tree in a network whose topology is *known*. Furthermore, our results hold for very general classes of graphs and in particular for every edge-density, in contrast to previous works which mainly concentrated on rings and cliques.

Our result enables one to prove an $\Omega(|E| + |V| \log |V|)$ lower bound on the communication complexity of any spanning tree construction algorithm, thus implying optimality of the algorithm of [GHS].

Some of the above results have been reported in an earlier version of this paper [AGV]. Results somewhat weaker than [AGV] have been independently obtained by [RK]. (The lower bound of [RK] does not hold if the size of the network is known.)

The rest of the paper is organized as follows. In Section 2 we define the model used for the main result and state the problem. In Section 3 we state and prove the lower bound on the message-complexity of broadcast in the model KT_1. In Section 4 we give the lower bound for the general model KT_ρ, and in Section 5 we present the upper bound.

2 The model

2.1 Basics

Our communication model consists of a point-to-point communication network, described by a simple undirected graph $G(V, E)$, where the vertices represent network processors and the edges represent bidirectional communication channels operating between them.

Whenever convenient, we will assume that $V = \{1, 2, ..., |V|\}$. Initially, (unique) ID's are assigned to the processors (vertices) of the graph G. These ID's are taken from an ordered set of integers $S = \{s_1, s_2, \ldots\}$ where $s_i < s_{i+1}$ for every $i \geq 1$. Thus a system configuration consists of a graph G and an ID-assignment, which is a one-to-one mapping $\phi : V \to S$.

One can distinguish between *synchronous* and *asynchronous* network models, as in [A1]. For the lower bound, we assume here that communication is synchronous; i.e., communication takes place in "rounds", where processors transmit only in the very beginning of a round and all messages are received by the end of the round. Clearly, the lower bound holds also if communication is asynchronous. For the upper bound, we assume that the network is asynchronous. Thus, our results hold assuming either synchronous or asynchronous communication.

A *protocol* is a local program executed by all the vertices in the network. In every step, each processor performs local computations, sends and receives messages and changes its local state according to the instructions of the protocol. A vertex starts executing a protocol either by means of a special *wake-up* signal, or as a result of receiving a message of the protocol. The set of vertices which can possibly receive a wake-up signal is called the *initiators* of the protocol. A protocol achieving a given task should work on every network G, and every assignment ϕ of ID's to the processors of G.

In order to enable a convenient way of measuring the size of messages, we introduce the following formalism. We assert that programs have local variables of two types: *identity* (*ID-typed*) variables $\bar{I} = (I_1, I_2, \ldots)$ and *ordinary* variables $\bar{X} = (X_1, X_2, \ldots)$. Initially, the ID-typed variables are empty, except for the ID-typed *input* variables (say, the first in the list of variables), which contain the ID's of some processors in some standard order. The ordinary variables initially contain some constants (e.g. 0 and 1). We want our lower bounds to apply also to the case where the size of the network is known to the processors, so we assume also that the ordinary input variable X_1 contains $|V|$. (This will not be used in our upper bound proofs.) The *state* of a processor v consists of the combined list, $L = (\bar{X}, \bar{I})$.

We assume that all messages sent by the protocol contain at most a constant number B of vertex ID's. (Alternatively, we could have allowed longer messages, but charged them by the number of processor ID's they contain.) Our complexity measure is the number of messages (containing at most B vertex ID's) sent in the worst-case execution of the protocol on the network $G(V, E)$.

Specifically, the communication instructions of the program are of two types: an unconditional "receive" message, and a (possibly) conditional "send" message. The condition in the "send" instruction is a comparison of two ordinary variables. (Note that this does not restrict generality, as allowing the condition to be a comparison of two ID-typed variables does not change the computational power of the protocol.) Messages consist of the values of some of the variables of the local program. Without loss of generality we may further assume that all "send" instructions are of the form

if $X_i = X_j$ *then* send the message $(I_{k_1}, I_{k_2}, \ldots, I_{k_B}; Z)$ to processor $I_{k_{B+1}}$.

This instruction sends the contents of B ID-typed variables plus an additional information field Z, to the processor whose ID is stored in $I_{k_{B+1}}$. The receiving processor may store some or all of the received values in its variables. In proving our lower bounds we will naturally have to be more specific about further restricting the allowed content of the additional information field Z.

We also assume, without loss of generality, that each processor can send at most one message to each of its neighbors in each round.

Finally let us give a precise statement of the problem of *broadcast from a single source*. One of the vertices is marked as *source*, and it has a certain *value*. The fact that a vertex is a source, and the value which needs to be broadcast is kept at a special input tape. This value should be disseminated from the source vertex to all vertices in the network, which will write it on their output tape.

2.2 Partial knowledge of the topology

The local program at a vertex has local *input* and local *output* variables. Our hierarchy of models KT_ρ (for $\rho \geq 0$) is characterized by the local inputs regarding the topology.

Definition: Denote the *distance* between two vertices $u, v \in V$ by $dist(u,v)$. For every $v \in V$ and $e = (u,w) \in E$ denote

$$dist(v,e) = \min\{dist(v,u), dist(v,w)\}.$$

Definition: In the model KT_ρ, the input to the local program at a vertex v contains all (and only) the edges e such that $dist(v,e) < \rho$ (where "storing" an edge means having a designated pair of variables holding the ID's of its endpoints).

In particular, in the anonymous model KT_0, no topological information is stored. In KT_1, a vertex knows all edges incident to itself, hence it knows the ID's of its neighbors. However, it does not know which pairs of its neighbors are connected by edges, since these edges are already at distance 1 from it. For general ρ, v knows almost all the subgraph of G induced by all vertices at radius ρ from v; the only "unknowns" are the (possible) edges connecting two vertices at distance exactly ρ from v.

We comment that the results for general ρ hold with only small changes if we use the more natural definition for KT_ρ, by which each vertex v simply knows the subgraph of G induced by all vertices at radius ρ from it. The only reason for defining the models in this particular way was to ensure the compatibility between the first two levels and the traditional models.

3 The lower bound for KT_1

Following some necessary definitions (given in Subsections 3.1 and 3.2), our lower bound proof proceeds in several stages. In the first stage (Subsections 3.3 through 3.5) we prove the claim only for $\rho = 1$, and only in a restricted model of *comparison protocols*. The proof is then extended (in Subsection 3.6) to the general model, which allows arbitrary computations at vertices. In Section 4 we handle the case of an arbitrary ρ.

We begin the section by giving some preliminary definitions and developing necessary tools. This is done in the first three subsections.

3.1 Executions, histories and similarity

Definition: We denote the *execution* of a protocol Π on a synchronous network $G(V,E)$ with an ID-assignment ϕ by $EX(\Pi, G, \phi)$. (This execution adheres to the rules of the standard synchronous model as described in Subsection 2.1; we omit a formal definition.) Denote the state of a processor v in the beginning of round i of the execution EX by $L_i(EX, v)$.

Definition: The *decoded representation* of a message sent during the execution $EX(\Pi, G, \phi)$ is obtained by replacing each ID value $\phi(v)$ occurring in the message by v.

Definition: The *message history* of an execution $EX = EX(\Pi, G, \phi)$, denoted $h(EX)$, is a sequence of quadruples

$$(ROUND, SENDER, RECEIVER, MESSAGE)$$

containing the messages sent during the execution in decoded representation. The quadruples are ordered lexicographically by the first three entries. The message history of a particular round i in the execution EX, denoted $h_i(EX)$, is the subsequence of $h(EX)$ with $ROUND = i$.

Definition: Consider a protocol Π, two graphs $G_0(V, E_0)$ and $G_1(V, E_1)$ over the same set of vertices V and two ID-assignments ϕ_0 and ϕ_1 for V, and the corresponding executions $EX_0 = EX(\Pi, G_0, \phi_0)$ and $EX_1 = EX(\Pi, G_1, \phi_1)$. We say that two messages M_1 and M_2 sent during these executions (respectively) are *similar* if their decoded representation is identical. Likewise, we say that the executions are *similar* if their message history is identical.

We state the following immediate fact for future use.

Fact 3.1: *Similarity of executions is transitive.* ▮

A crucial element of our lower bound proof involves finding pairs of ID-assignments ϕ_0, ϕ_1 whose substitution in the processors of the network essentially "preserves" the execution. We now formalize this notion of "mixable" ID-assignments.

Definition: Let ϕ_0 and ϕ_1 be two ID-assignments with disjoint ranges. For any n-bit string $\bar{\alpha} = \alpha_1 \ldots \alpha_n$, let $\phi_{\bar{\alpha}}$ be an ID-assignment such that for every $i \in V$

$$\phi_{\bar{\alpha}}(i) = \phi_{\alpha_i}(i).$$

The *mixing set* of ϕ_0 and ϕ_1 is defined as

$$\mathcal{M}(\phi_0, \phi_1) = \{\phi_{\bar{\alpha}} \mid \bar{\alpha} \in \{0,1\}^n\}.$$

We say that the ID-assignments ϕ_0 and ϕ_1 are *fully mixable* for the protocol Π and the graph $G(V, E)$ if all the executions $EX(\Pi, G, \phi_{\bar{\alpha}})$ (for every $\phi_{\bar{\alpha}}$ in the mixing set $\mathcal{M}(\phi_0, \phi_1)$) are similar.

3.2 Edge utilization, charge-counts and message complexity

The goal of this Subsection is to establish some definitions which will enable us to effectively bound the message complexity of our broadcast algorithms.

Definition: We say that an edge $(u, v) \in E$ is *utilized* during an execution $EX(\Pi, G, \phi)$ if at least one of the following three events takes place:

(i) A message is sent on (u, v).

(ii) Processor u either sends or receives a message containing $\phi(v)$.

(iii) Processor v either sends or receives a message containing $\phi(u)$.

This definition provides us with an accounting method for charging messages sent during the execution of any protocol to the links of the network.

Definition: The *charge count* of an execution $EX(\Pi, G, \phi)$ is obtained by employing the following *charging rule*. For every message containing $\phi(z)$ that is sent during the execution from the processor x to the processor y, we charge

(C1) the edge (x, y),

(C2) the pair (possibly edge) (x, z), and

(C3) the pair (possibly edge) (z, y).

We now claim that the above charge-count, the number of messages sent during the execution and the number of edges utilized during the execution are closely related, and particularly, the message complexity of the execution is at least a (positive) constant factor times the number of utilized edges. We stress that this does *not* imply that messages *must actually be sent over every utilized edge.*

A message sent from x to y is charged to the edge (x, y) and to all the pairs (x, z) and (y, z) such that $\phi(z)$ occurs in the message. Since there are at most B ID's in each message, we have the following lemma.

Lemma 3.2: *The number of edges that get charged for a single message sent during an execution $EX(\Pi, G, \phi)$ is at most $2B + 1$.* ▮

Also, inspection of the definitions of edge utilization and charge-count reveals the following.

Lemma 3.3: *Applying the charge count of the above definition to an execution $EX(\Pi, G, \phi)$, each utilized edge gets charged at least once.* ▮

Lemma 3.4: *Let m denote the number of utilized edges in an execution $EX(\Pi, G, \phi)$. Then the message complexity of the execution is $\Omega(m)$.*

Proof: Let C denote the total charge placed by the above rules on the execution $EX(\Pi, G, \phi)$, and let M denote the total number of messages sent during that execution. Combining Lemma 3.2 with Lemma 3.3, we get

$$m \leq C \leq (2B + 1)M.$$

Recalling that B is a constant, the Lemma follows. ▮

3.3 The model of comparison protocols

At this point, we restrict the local computations of the program which involve processors' ID's to comparing two ID's. The local computations of the program may involve operations of the following two types:

1. Comparing two ID-typed variables I_i, I_j and storing the result of the comparison in an ordinary variable. We refer to this result as $comp(I_i, I_j)$. Since the set S of possible ID's is ordered, the result of the comparison may be either of the three values "$<$", "$=$" or "$>$". We assume some standard encoding of the result of the comparison. For instance, encode "$<$" as -1, "$=$" as 0 and "$>$" as $+1$.
2. Performing an arbitrary computation on ordinary variables and storing the result in another ordinary variable.

Under this restriction, our "send" instructions may be allowed to include the entire list of ordinary variables of the sending processor in their "additional information" field Z.

The reason for restricting the permissible operations of local programs on ID's to comparisons is that this makes it easy to prove the existence of a fully mixable ID-assignment.

Definition: Two lists of ID's $\bar{a} = (a_1, \ldots, a_k)$ and $\bar{b} = (b_1, \ldots, b_k)$, $\bar{a}, \bar{b} \subseteq S$ are called *adjacent* if the following conditions hold:

- For every $1 \leq i \leq n$ and for every $s \in S - \{a_i, b_i\}$, $comp(a_i, s) = comp(b_i, s)$.
- For every $1 \leq i, j \leq n$, $comp(a_i, a_j) = comp(b_i, b_j)$.

Two *ID-assignments* ϕ_0, ϕ_1 are *adjacent* if the corresponding lists are adjacent (where the list corresponding to an ID-assignment ϕ is $(\phi(1), \ldots, \phi(|V|))$.

Clearly, if $|S| \geq 2|V|$ then there exist two adjacent assignments ϕ_0, ϕ_1 with disjoint range. (For example, let $\phi_i(j) = s_{2j-i}$.)

Lemma 3.5: *Let ϕ_0 and ϕ_1 be two adjacent ID-assignments. Then for any protocol Π and any graph $G(V, E)$, the executions $EX(\Pi, G, \phi_0)$ and $EX(\Pi, G, \phi_1)$ are similar.*

Proof: Immediate by observing that all the ordinary variables of all local programs have the same values in both executions. ▮

Corollary 3.6: *Let ϕ_0 and ϕ_1 be adjacent ID-assignments with disjoint ranges. Then ϕ_0 and ϕ_1 are fully mixable for any protocol Π and any graph $G(V, E)$.* ▮

3.4 Networks and ID-assignments for the lower bound

We are now ready to introduce the family of networks to be used in our lower bound proof. Consider a graph $G(V, E)$ with a specially designated source s. Construct a graph $G'(V', E')$ where

$$V' = \{w' \ : \ w \in V\},$$
$$E' = \{(w'_1, w'_2) \ : \ (w_1, w_2) \in E\}$$

and G' has no special source vertex. Construct a graph $G^2(V^2, E^2)$ where

$$V^2 = V \bigcup V'$$

and

$$E^2 = E \bigcup E'.$$

Thus $G^2(V^2, E^2)$ is the graph consisting of two (disconnected) identical copies of G, one with a source and one without a source.

For every edge $e = (u, v) \in E$, let $e' = (u', v')$ and construct $G^e(V^2, E^e)$ by letting

$$E^e = (E - \{e\}) \bigcup (E' - \{e'\}) \bigcup \{(u, v'), (u', v)\}.$$

Namely, G^e consists of two identical copies of $G - \{e\}$ connected with corresponding crossing edges (i.e., u of one copy to v of the other, and vice versa). Now u' has the same topological environment as u.

Example: Consider the graph $G(V, E)$ where $V = \{1, 2, 3\}$ and $E = \{(1, 2), (1, 3), (2, 3)\}$, and the edge $e = (1, 3)$. The corresponding graphs G^2 and G^e are presented in Figures 1 and 2, respectively. ∎

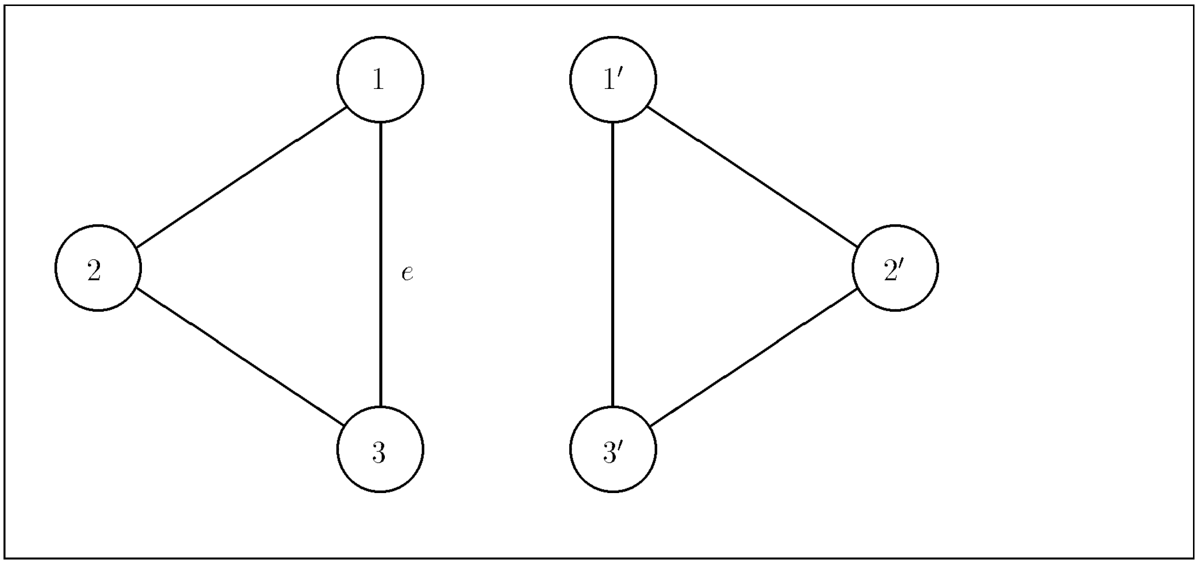

Figure 1: The graph $G^2(V^2, E^2)$ corresponding to the example

Further, for any given graph $G(V, E)$ define the family of graphs

$$C_G = \{G^e : e \in E\}.$$

Note that G and G^2 are *not* included in C_G. Also note that all the graphs in C_G have the same number of vertices (i.e. $|V^2| = 2|V|$) and the same number of edges (i.e. $|E^e| = 2|E|$). Therefore the variable X_1, storing the size of the network, will contain the same value in the run of any algorithm on any of these graphs.

For the remainder of the section we fix $G(V, E)$ to be some arbitrary graph and construct the class C_G. We fix some specific vertex $s \in V$ as the source in all the graphs of C_G. (Note that s occurs in the first copy of G in all of these graphs.)

Let S be a set of ID's ($|S| \geq 2|V|$), and let $\phi_0 : V \to S$ and $\phi_1 : V \to S$ be two adjacent ID-assignments with disjoint ranges. By Corollary 3.6, ϕ_0 and ϕ_1 are fully mixable for any protocol Π and any graph G on the vertex set V or V' (taking $\phi_i(v') = \phi_i(v)$ for $v \in V$, where v' is the corresponding vertex in V'). Define the ID-assignment $\psi : V^2 \to S$ as $\psi(w) = \phi_0(w)$ and $\psi(w') = \phi_1(w)$ for every $w \in V$. Let Π be a protocol that achieves broadcast from s on at least a fraction $\delta > 0$ of the graphs of C_G with the ID-assignment ψ.

Our goal is to prove that for a constant fraction of graphs in C_G, Π requires $\Omega(|E|)$ messages. In our lower bound argument we concentrate on the graph G^2, switching whenever required to one of the auxiliary graphs G^e, and relying on the fact that the protocol Π is correct when run on $G^e \in C_G$. We argue that neighbors must "hear" of one another during any execution of a broadcast protocol, and therefore $\Omega(|E|)$ edges need to be utilized. The intuition behind the proof is that in case some edge $e \in E$ is not utilized, no processor in the network can distinguish the case in which it takes

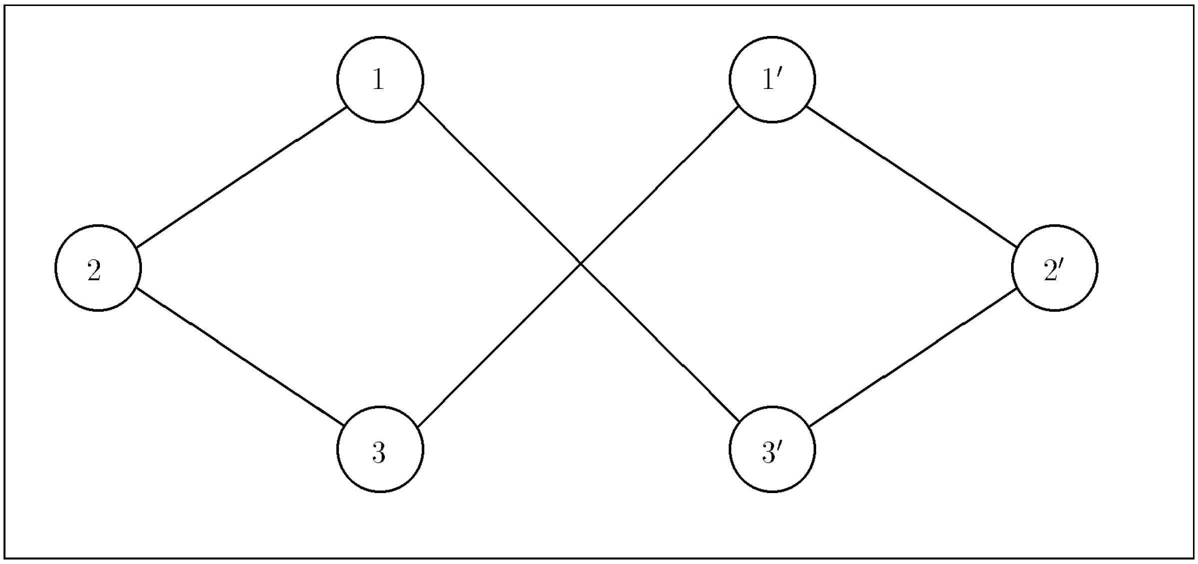

Figure 2: The corresponding graph $G^e(V^2, E^e)$

part in an execution on G^2 from the case in which it takes part in an execution on G^e. The only potential difference between these executions lies in whether u and v are neighbors or not, where $e = (u, v)$. But this neighborhood relation can not be tested if no messages bearing the ID of one processor are communicated from/to the other.

This intuition needs careful formalization, which requires us to define some appropriate ID-assignments and executions for the graphs of C_G. For every $e = (u, v) \in E$, define the ID-assignment $\psi_u^e : V^2 \to S$ just as ψ, except for interchanging the ID's of u and u' (i.e., letting $\psi_u^e(u) = \phi_1(u)$ and $\psi_u^e(u') = \phi_0(u)$), and define ψ_v^e analogously for v. Finally define

$$
\begin{aligned}
EX &= EX(\Pi, G^2, \psi), \\
EX_u^e &= EX(\Pi, G^2, \psi_u^e), \\
EX_v^e &= EX(\Pi, G^2, \psi_v^e), \\
EX^e &= EX(\Pi, G^e, \psi).
\end{aligned}
$$

3.5 The lower bound proof

We start by observing the following lemma.

Lemma 3.7: *The executions EX, EX_u^e and EX_v^e are similar.*

Proof: The claim follows directly from the following facts. First, G^2 is composed of two completely disconnected graphs G and G'. Secondly, ϕ_0 and ϕ_1 are fully mixable for Π and G as well as for Π and G'. Finally, ϕ_0, ϕ_1 and the parts of ψ_u^e and ψ_v^e restricted to G and to G' are in the mixing set $\mathcal{M}(\phi_0, \phi_1)$. ∎

We need to argue about the relationships between these executions and the execution on G^e, EX^e.

Lemma 3.8: *Suppose that during the first r rounds of the execution EX both e and e' are not utilized, for some $e = (u, v) \in E$. Then the following hold for every round $1 \le i \le r$ of the executions EX, EX_u^e, EX_v^e and EX^e:*

(1) *The states of the processors in the beginning of the round satisfy:*

- **(1.1)** *For every processor $w \in V^2 - \{u, u', v, v'\}$, $L_i(EX^e, w) = L_i(EX, w)$.*
- **(1.2)** *For $x \in \{u, u'\}$, $L_i(EX^e, x) = L_i(EX_v^e, x)$.*
- **(1.3)** *For $x \in \{v, v'\}$, $L_i(EX^e, x) = L_i(EX_u^e, x)$.*

(2) *The messages sent during the round are similar, i.e., $h_i(EX) = h_i(EX_u^e) = h_i(EX_v^e) = h_i(EX^e)$.*

(3) *In EX^e, no messages are sent during the round over the edges (u, v') and (v, u').*

Proof: Let us first remark that the assumption that e is not utilized in round i of EX implies that it is not utilized in EX_u^e or EX_v^e either, since these executions are similar (and the graphs are identical).

As a major step towards proving the lemma we argue the following.

Claim 3.8.1: *If (1) holds at the beginning of a round $1 \le i \le r$ then (2) and (3) must hold during the round.*

Proof: By cases corresponding to the cases of (1). In general, for each processor w we show that the messages sent by it in the execution EX^e are similar to those sent by it in *some* of the other three (similar) executions. This suffices in order to prove (2) due to the transitivity of similarity.

Let us first consider any processor $w \in V^2 - \{u, u', v, v'\}$. Part (1.1) ensures that w sends precisely the same messages in round i of EX and EX^e, since its state is identical. We need to show that these messages are not only identical but also similar (i.e., they are identical also in decoded representation). This is immediate since both executions use the same ID-assignment, ψ. Consequently, the part of (2) that is relevant to w follows.

Next, consider the processor u. Part (1.2) ensures that its states in the beginning of the round in executions EX_v^e and EX^e are identical, and therefore u executes the same "send" instructions and sends precisely the same messages in both runs. Again we need to show that these messages are not only identical but also similar. This requires us to show that every ID value sent in these two executions by u has the same meaning (i.e., it represents the same processor) under ψ and ψ_v^e. The assumption that e is not utilized in round i of EX_v^e implies that the particular ID-typed variable I storing $\phi_1(v)$ in u is not used in any "send" instruction executed by u in the execution EX_v^e, neither in its content nor in its destination field. Consequently in EX^e this variable is not used either. Since in G^2 the two copies of the graph G are disconnected, any ID held by u in the execution EX_u^e is the ID of some $w \in V$. Every processor in $V - \{v\}$ has the same ID under ψ and ψ_v^e. Consequently, the part of (2) that is relevant to u follows.

Since I is not used as a destination field of any "send" instruction, no message is sent from u to v' in this round of EX^e, which accounts for the part of (3) that is relevant to u.

The case of u' is handled in the same way. As for the cases of v and v', these are handled analogously, using (1.3) in place of (1.2) and discussing EX_u^e instead of EX_v^e. This completes the proof of Claim 3.8.1. ∎

We now prove the lemma by induction on i, the round number. For the induction base, $i = 1$, Part (1) follows from the definition of the input variables of processors under a given topology and ID-assignment, and (2) and (3) follow by Claim 3.8.1.

For the induction step we assume that (1), (2) and (3) hold during rounds $0, \ldots, i-1$, $i \le r$, and look at round i. By induction hypothesis, the messages sent during the $i-1$-st round of all four executions are similar. A case analysis converse to that of the proof of Claim 3.8.1 establishes that (1) holds at the beginning of round i. For instance consider the processor u. The messages it gets in the end of round $i - 1$ are similar in EX_v^e and EX^e. It does not get a message on the edge e in EX_v^e or on the edge (u, v') in EX^e. Also it doesn't get a message containing the ID of v in either execution (since otherwise the executions cannot be similar). Consequently all the messages it gets contain only ID's of vertices from $V - \{v\}$. For these processors, the ID's assigned by ψ_v^e and ψ are identical, hence any received values that are stored by u are identical in both executions (relying on the fact that any local computations made by u in order to determine which values to store will again be identical by the inductive assumption). Similar arguments apply for the other processors. Parts (2) and (3) follow by Claim 3.8.1. This completes the proof of the lemma. ∎

Corollary 3.9: *Suppose that during the execution EX both e and e' are not utilized, for some $e = (u, v) \in E$. Then the executions EX and EX^e are similar ($h(EX) = h(EX^e)$), and furthermore, in EX^e no messages are sent over the edges (u, v') and (v, u').* ∎

Lemma 3.10: *Suppose that for some $e = (u, v) \in E$, both e and e' are not utilized during the first r rounds of the execution EX, but e or e' (or both) is utilized in round $r + 1$ of EX. Then for every other edge $e_1 \in E - \{e\}$, if e_1 is utilized in round $r + 1$*

of EX then it is utilized also in EX^e.

Proof: We first note that Part (1) of Lemma 3.8 holds also for round $r+1$, by the same inductive proof. Repeating an analysis similar to that in the proof of Claim 3.8.1 we can show that the "send" instructions executed by each processor during round $r+1$ are identical in EX^e and the appropriate execution according to the cases of Part (1) of Lemma 3.8. The difference, however, is that in this round, the edge e *does* get utilized, hence two corresponding messages may contain the ID's of different processors. Nonetheless, one can see that $h_{r+1}(EX)$ is still almost identical to $h_{r+1}(EX^e)$, and the only possible discrepancies are the following:

- the processor u might send $\phi_0(v)$ (the ID of v) in EX and $\phi_1(v)$ (the ID of v') in EX^e,
- the processor u' might send $\phi_1(v)$ in EX and $\phi_0(v)$ in EX^e,
- the processor v might send $\phi_0(u)$ in EX and $\phi_1(u)$ in EX^e,
- the processor v' might send $\phi_1(u)$ in EX and $\phi_0(u)$ in EX^e.

A straightforward case analysis shows that the only edge whose utilization may be affected by these discrepancies is e, and for every other edge $e_1 \in E - \{e\}$, if e_1 is utilized in this round of EX then it is utilized also in EX^e. ∎

Lemma 3.11: *For at least a fraction δ of the edges $e \in E$, either e or e' is utilized in the execution EX.*

Proof: Suppose otherwise. By Corollary 3.9, more than a fraction δ of the edges $e = (u,v) \in E$ satisfy the condition that, in EX^e, no messages are sent over the edges (u,v') and (v,u'). But then, in EX^e, the broadcast message never reaches u' or v', so Π performs incorrectly on more than a fraction δ of the graphs in C_G, under ID-assignment ψ. ∎

Let E_i denote the set of edges $e \in E$ such that e or e' is utilized during the first i rounds of EX, and let $Q_i = E - E_i$. Consider the first round r such that $|E_r| \geq \delta|E|/2$ (such a round exists by Lemma 3.11). Consider any edge $e = (u,v) \in Q_{r-1}$. By Lemma 3.8, the first $r-1$ rounds of the executions EX and EX^e are similar. Furthermore, by Lemma 3.10 in the r'th round the histories are either identical (if also $e \in Q_r$) or almost identical, disagreeing only in some occurrences of $\phi_0(u)$, $\phi_1(u)$, $\phi_0(v)$ and $\phi_1(v)$ (if e is utilized in round r). Thus all the edges in E_r except possibly e are utilized during the first r rounds of EX^e.

Lemma 3.12: *For every $e \in Q_{r-1}$, the message complexity of the execution EX^e is $\Omega(\delta|E|)$.*

Proof: By direct application of Lemma 3.4, noting that m, the number of utilized edges, satisfies $m = \Omega(\delta|E|)$, since $m \geq |E_r| \geq \delta|E|/2$. ▮

Observe that, by definition, $|Q_{r-1}| \geq (1 - \frac{\delta}{2})|E|$. Therefore the lower bound implied by Lemma 3.12 applies to a constant fraction of the networks in C_G. This gives us our theorem.

Theorem 3.13: *Let $G(V, E)$ be an arbitrary graph, and let Π be a protocol with any set of initiators achieving broadcast on at least a fraction $\delta > 0$ of the networks of the family C_G in the comparison model. Then the message complexity of Π is $\Omega(\delta|E|)$ on a constant fraction of the networks of C_G. This holds even when the vertices know the size of the network.* ▮

This strong formulation of the Theorem enables us to extend the result and derive a statement concerning randomized protocols as well.

Theorem 3.14: *Let $G(V, E)$ be an arbitrary graph, and let Π be a randomized (Monte-Carlo) protocol with any set of initiators achieving broadcast on the networks of the family C_G (in the comparison model) with error probability less than ϵ. Then the average message complexity of Π on the networks of C_G is $\Omega((1-\epsilon)|E|)$. This holds even when the vertices know the size of the network.*

Proof: View the randomized protocol Π as a probability measure μ over a collection $\{\pi_i\}$ of deterministic protocols; in every execution one of these protocols is randomly selected and used according to μ.

Let $\delta_G(\pi)$ be the fraction of graphs in C_G on which the deterministic protocol π achieves broadcast. Since Π errs with probability less than ϵ, $\int_\Pi \delta_G d\mu > 1 - \epsilon$. By Theorem 3.13, the average message complexity of Π over C_G is at least $\int_\Pi c|E|\delta_G d\mu > c(1-\epsilon)|E|$, for some constant $c > 0$.

Note that this result is tight, since a Monte-Carlo broadcast algorithm can begin with the source deciding, with probability $1 - \epsilon$, to initiate a flooding algorithm, and with probability ϵ to do nothing. The average message complexity of this algorithm is $O((1-\epsilon)|E|)$. ▮

3.6 Extending the proof to the unrestricted model

We now extend the result of Theorem 3.13 by getting rid of the simplifying restrictions imposed on the local programs in the comparison model, and allowing arbitrary computations in the local programs. This introduces difficulties unencountered so far. We have to explicitly bound the length of messages, otherwise an unbounded number of ID's can be transferred in a single message. Also, we have to disallow protocols with time unbounded in terms of the network topology, otherwise one may encode an unbounded number of ID's by the choice of transmission round. (This clearly relates only to the synchronous communication model. In the asynchronous model such encoding is impossible!)

We introduce an upper bound T on number of rounds, and an upper bound L on the length of the "additional information" field Z of messages, both depending only on G. Modifying the argument of Section 3.5, we get

Theorem 3.15: *Let $G(V,E)$ be an arbitrary graph, and Π a protocol with arbitrary local computations achieving broadcast on every network of the family C_G. Further, assume that Π requires at most T rounds, and the length of the "additional information" field Z of its messages is at most L, where both bounds depend only on G. Then the message complexity of Π is $\Theta(|E|)$ on a constant fraction of the networks of C_G.*

Proof: The only way in which we used the restriction to comparison models was in proving the existence of ID-assignments ϕ_0 and ϕ_1 that are fully mixable for Π with respect to both G and G'. In the general model we can not use the "adjacency" property which is now irrelevant. Instead, we use a Ramsey Theory argument [GRS].

Let S be an arbitrary ordered set of ID's. Denote $n = |V|$ and $m = |E|$. Let ξ denote the number of possible nonsimilar executions $EX(\Pi, G^2, \phi)$ (i.e., executions with different message histories) over all possible ID-assignments $\phi : V \to S$ satisfying $\phi(u) < \phi(v)$ for every $u < v$. The number of possible messages (in decoded representation) is at most $(2n)^B \cdot 2^L$. Assuming that less than m messages are sent (otherwise the Theorem holds trivially), one can readily verify that the number of possible quadruples

$$(ROUND, SENDER, RECEIVER, MESSAGE)$$

(where messages are in decoded representation) is bounded above by $T \cdot 4m \cdot (2n)^B \cdot 2^L$. It follows that

$$\xi < \binom{T \cdot 4m \cdot (2n)^B \cdot 2^L}{m}.$$

The execution histories (or, the equivalence classes of similar executions) induce a ξ-coloring of the n-subsets of S. Ramsey's Theorem asserts that if S is sufficiently large (though still finite) then there exists a set $R \subset S$ of $2n$ ID's so that all the n-subsets of R have the same color [GRS, Sec. 1.2]. The set R is then partitioned into two disjoint subsets $R_0 = \{r_i^0 \mid 1 \le i \le n\}$ and $R_1 = \{r_i^1 \mid 1 \le i \le n\}$, and the desired ID-assignments are defined as $\phi_0(i) = r_i^0$ and $\phi_1(i) = r_i^1$ for every $1 \le i \le n$. By choice, these ID-assignments are fully mixable. The rest of the proof follows as in the restricted case.

A simpler (but more tedious) direct argument is possible, noting that we do not really need a collection of monochromatic n-subsets of a $2n$ elements set, but rather a collection of $n+1$ monochromatic subsets $S_0, S_1, ..., S_n$ such that $|S_0 \bigcap S_i| = n-1$, for all $1 \le i \le n$. ∎

We remark that we do not know how to extend the result for general randomized protocols, since the Ramsey argument might produce a different pair of fully mixable ID-assignments for each of the deterministic algorithms in the collection $\mathcal{R}$ constituting the randomized protocol. However, it is possible to derive the result for restricted randomized protocols in which there is a bound on the number of "coinflips," or, the size of the collection $\mathcal{R}$, and this bound is independent of the ID's. For such protocols we can apply the above technique to produce a single pair of ID assignments that works for all the deterministic algorithms in $\mathcal{R}$ simultaneously, since a finite bound on ξ still exists.

4 Lower bound for the model of partial topological knowledge

In order to prove Theorem 2, giving the lower bound for the model KT_ρ, for any $\rho \geq 2$, we need to go through the entire proof and revise it to this more powerful setting. The problem is that in order for a vertex u to distinguish between the original graph G^2 and some "switched graph" G^e, $e = (x, y)$, it does not have to be incident to the edge, or to get the ID of some endpoint; it is enough that it gets the ID of some vertex w in distance ρ or less from itself which is "on the other side of e" (i.e., such that some short path from v to w goes through e). The definitions of edge utilization and charge count from Section 3.2 have to be modified accordingly.

In order to define our graph family we need the following result. Let $g(G)$ denote the *girth* of a graph G, i.e., the length of a smallest cycle in G. (A single edge is not considered a cycle of length 2, so $g(G) \geq 3$ for every G.)

Proposition 4.1 [Bo]: *There exists a constant $c > 0$, such that for every integer $\rho \geq 2$ there exists a graph $G(V, E)$ with girth $g(G) > 2\rho$ and $|E| = \Omega(|V|^{1+\frac{c}{\rho}})$.*

Our family of graphs, F_ρ, is constructed as follows. We first pick a graph $G(V, E)$ satisfying the conditions of Proposition 4.1, and then let $F_\rho = C_G$.

The advantageous property of the networks in C_G is that for every two vertices of distance at most ρ from each other, there is a unique path of that length connecting them.

Definition: In a graph G with girth $g(G) > 2\rho$, two vertices $u, v \in V$ are said to be *e-connected* for some edge $e \in E$ if they are at distance at most ρ and the (unique) shortest path between them contains e.

Note that for every two processors u and v, the number of edges e such that u and v are e-connected is at most ρ (in fact, this number is exactly the distance between them if this distance is at most ρ, and 0 otherwise).

Definition: We say that an edge $e = (u, v) \in E$ is *utilized* during an execution $EX(\Pi, G, \phi)$ if at least one of the following events takes place:

(i) A message is sent on (u, v).

(ii) Processor x either sends or receives a message containing $\phi(z)$, for two e-connected processors $x, z \in V$.

Definition: The *charge count* of an execution $EX(\Pi, G, \phi)$ is obtained by employing the following *charging rule*. For every message containing $\phi(z)$ that is sent from the processor x to the processor y, we charge

(C1) the edge (x, y), and

(C2) every edge e such that z is e-connected to either x or y.

Lemma 4.2: *For every graph G with girth $g(G) > 2\rho$, the number of edges that get charged for a single message sent during an execution $EX(\Pi, G, \phi)$ is at most $2B\rho + 1$.*

Proof: A message sent from x to y is charged to the edge (x, y). In addition, for every ID $\phi(z)$ occurring in the message, the message is charged to all the edges e such that either x and z or y and z are e-connected. Since G has girth greater than 2ρ, there are at most ρ edges of each of these types. Since there are B ID's in each message, the Lemma follows. ∎

Lemma 4.3: *Applying the charge count of the above definition to an execution $EX(\Pi, G, \phi)$, each utilized edge is charged at least once.*

Proof: For each utilized edge $e = (u, v)$ consider the following three cases:

(i) A message is sent on e: Then e is charged by (C1).

(ii) Processor x either sends or receives a message containing $\phi(z)$, for two e-connected processors $x, z \in V$: Then e is charged by (C2).

The Lemma follows. ∎

Lemma 4.4: *Let G be a graph with girth $g(G) > 2\rho$, and let m denote the number of utilized edges in an execution $EX(\Pi, G, \phi)$. Then the message complexity of the execution is $\Omega(m/\rho)$.*

Proof: Let C denote the total charge placed by the above rules on the execution $EX(\Pi, G, \phi)$, and let M denote the total number of messages sent during that execution. Combining Lemma 4.2 with Lemma 4.3, we get

$$m \leq C \leq (2B\rho + 1)M.$$

Recalling that B is a constant, the Lemma follows. ∎

Definition: For every edge $e = (u,v) \in E$, the neighborhood $\Gamma_i(u,e)$ is the i-neighborhood of u (i.e., the set of nodes at distance i from u) in the graph obtained from G by eliminating the edge e.

For proving the lower bound we define the graphs G^2 and G^e (for every $e = (u,v) \in E$), the ID-assignment ψ and the executions EX and EX^e as before. We need to define a collection of "intermediate" ID-assignments $\psi^e_{u,j}$ and $\psi^e_{v,j}$, for $1 \le j \le \rho$, as follows: define $\psi^e_{u,j}$ just as ψ, except for interchanging the ID's of w and w' for every $w \in \Gamma_j(u,e)$, and define $\psi^e_{v,j}$ analogously for v. Define the executions $EX^e_{u,j}$ and $EX^e_{v,j}$ accordingly. The main lemma parallel to Lemma 3.8 becomes:

Lemma 4.5: *Suppose that during the first r rounds of the execution EX both e and e' are not utilized, for some $e = (u,v) \in E$. Then the following hold for every round $1 \le i \le r$ of the executions EX, EX^e and $EX^e_{u,j}$ and $EX^e_{v,j}$ for every $1 \le j \le \rho$:*

(1) *The states of the processors in the beginning of the round satisfy:*

- **(1.1)** *For every processor $w \in V^2 - \{u,u',v,v'\}$, $L_i(EX,w) = L_i(EX^e,w)$.*
- **(1.2)** *For $x \in (\Gamma_j(u,e) - \Gamma_{j-1}(u,e)) \cup (\Gamma_j(u',e) - \Gamma_{j-1}(u',e))$ (for $1 \le j \le \rho$), $L_i(EX^e_{\rho-j-1,v},x) = L_i(EX^e,x)$.*
- **(1.3)** *For $x \in (\Gamma_j(v,e) - \Gamma_{j-1}(v,e)) \cup (\Gamma_j(v',e) - \Gamma_{j-1}(v',e))$ (for $1 \le j \le \rho$), $L_i(EX^e_{\rho-j-1,u},x) = L_i(EX^e,x)$.*

(2) *The messages sent during the round are similar, i.e., $h_i(EX) = h_i(EX^e_{u,j}) = h_i(EX^e_{v,j}) = h_i(EX^e)$ for every $1 \le j \le$ rho.*

(3) *In EX^e, no messages are sent over the edges (u,v') and (v,u').*

The proof of this lemma follows arguments similar to those proving Lemma 3.8, although the overall proof becomes more complex. The rest of the proof also mimics the arguments of Section 3, and we omit the details. We have

Theorem 4.6: *There exists a constant $c' > 0$ such that for every two integers $\rho \ge 1$ and $n \ge 1$ there exists a family F_ρ of graphs with m edges and n vertices each, where $m = \Omega(n^{1+\frac{c'}{\rho}})$, such that any protocol that works correctly on all graphs in F_ρ in the model KT_ρ sends at least $\Omega(m/\rho)$ messages over a constant fraction of the graphs from F_ρ. This lower bound holds even if the network is synchronous, all the vertices start the protocol at the same round, and the size of the network is known.* ∎

5 The upper bound

In this section we prove Theorem 3, that is, we show that for any integer $\rho \geq 1$ and for any connected graph $G(V,E)$, in the model KT_ρ broadcast can be performed with at most $O(\min\{|E|, |V|^{1+\frac{c}{\rho}}\})$ messages for some constant $c > 0$. This upper bound holds even if the network is asynchronous.

Definition: A cycle is *short* if its length is 2ρ or less.

The key observation behind the algorithm is that if a vertex knows all the edges at distance ρ or less from it, then it can detect all short cycles going through it. This enables us to disconnect all short cycles locally, by deleting the heaviest edge (the one with the highest weight) in each such cycle.

More precisely, assume some (locally computable) assignment of distinct weights to the edges. Define a subgraph $\bar{G}(V,\bar{E})$ of G by marking the heaviest edge in every short cycle "unusable" and including precisely all unmarked edges in $\bar{E}$. We require only the vertices incident to an edge e to know whether or not e is usable. Therefore, given the partial topological knowledge of the vertices, such edge deletions can be performed locally by the vertices incident to each edge, without sending a single message.

Lemma 5.1: *If G is connected then $\bar{G}$ is connected as well.*

Proof: The claim holds even if one deletes the heaviest edge in *every* cycle of the graph. In fact, the remaining subgraph in such a case is a spanning tree of the original graph (cf. [E]). ▮

An immediate consequence of the marking process used to define $\bar{G}$ is that all short cycles are disconnected, and hence we have

Lemma 5.2: *The girth of $\bar{G}$ satisfies $g(\bar{G}) \geq 2\rho + 1$.* ▮

We need the following proposition.

Proposition 5.3 [A1, PS]: *There exists a constant $c > 0$ such that for any graph $G(V,E)$ and for any $k \geq 1$ there exists a subgraph $G'(V,E')$ such that*

1. *$|E'| \leq O(|V|^{1+\frac{c}{k}})$.*
2. *For every edge $(u,v) \in E$, the distance between u and v in G' is at most k. (I.e., G' is a k-spanner of G [PS].)*

Corollary 5.4: *There exists a constant $c > 0$ such that any graph $G(V,E)$ with girth $g(G) \geq 3$ has at most $|E| = O(|V|^{1+\frac{c}{g(G)-2}})$ edges.*

Proof: Select c as in Proposition 5.3 and set $k = g(G) - 2 \geq 1$. We claim that the only k-spanner of G (i.e., the only possible subgraph G' satisfying property 2 of Proposition 5.3) is G itself. To see this, consider any *proper* subgraph G' of G and let (u, v) be some edge in $E - E'$. The shortest path between u and v in G' is of length at least $g(G) - 1 = k + 1$, violating the desired property. Consequently, Proposition 5.3 implies that G itself satisfies also the first property, namely, $|E| = O(|V|^{1+\frac{c}{g(G)-2}})$. ∎

It follows from Lemma 5.2 and Cor. 5.4 that $|\bar{E}| = O(|V|^{1+\frac{c}{\rho}})$. We can now perform broadcast on $\bar{G}$ using by the standard flooding algorithm described earlier. That is, whenever a vertex receives the message for the first time, it sends it over all the usable edges $e \in \bar{E}$ incident to it. This requires $O(|\bar{E}|) = O(|V|^{1+\frac{c}{\rho}})$ messages. This completes the proof of our first Theorem.

Theorem 5.5: *There exists a constant $c > 0$ such that for every integer $\rho \geq 1$ and for any graph $G(V, E)$, broadcast can be performed in the model KT_ρ using at most $O(\min\{|E|, |V|^{1+\frac{c}{\rho}}\})$ messages. This upper bound holds even if the network is asynchronous.* ∎

ACKNOWLEDGMENTS

We wish to thank Mike Luby, Yishay Mansour, Rüdiger Reischuk, Avi Wigderson and two anonymous referees for their help in various stages of this work.

References

[A1] B. Awerbuch, "Complexity of Network Synchronization", *J. of the ACM*, Vol. 32, No. 4, 1985, pp. 804-823.

[AG] Y. Afek and E. Gafni, "Time and Message Bounds for Election in Synchronous and Asynchronous Complete Networks", *Proc. 4th Symp. on Principles of Distributed Computing*, 1985, pp. 186-195.

[AGV] B. Awerbuch, O. Goldreich and R. Vainish, "On the Message Complexity of Broadcast: a Basic Lower Bound", *Technical memo*, MIT/LCS/TM-325, April 1987.

[AAHK] K. Abrahamson, A. Adler, L. Higham and D. Kirkpatrick, "Probabilistic Solitude Verification on a ring", *Proc. 5th Symp. on Principles of Distributed Computing*, 1986, pp. 161-173.

[ASW] C. Attiya, M. Snir and M. Warmuth, "Computing on Anonymous Ring", *Proc. 4th Symp. on Principles of Distributed Computing*, 1985,pp. 196-203.

[Bo] B. Bollobas, *Extremal Graph Theory*, Academic Press, 1978.

[Bu] J.E. Burns, "A Formal Model for Message Passing Systems", TR-91, Indiana University, (1980).

[DM] Y.K. Dalal and R. Metcalfe, "Reserve Path Forwarding of Broadcast Packets", *Comm. ACM*, Vol. 21, No. 12, pp. 1040-1048, 1978.

[E] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[F] G.R. Frederickson, "Trade-offs for selection in distributed networks" *Proc. 2nd Symp. on Principles of Distributed Computing*, 1983, pp. 154-160.

[FL] G.R. Frederickson and N.A. Lynch, "The Impact of Synchronous Communication on the Problem of Electing a Leader in a Ring", *Proc. 16th ACM Symp. on Theory of Computing*, 1984, pp. 493-503.

[GI] E. Gafni and A. Itai, private communication, 1987.

[GHS] R.G. Gallager, P.A. Humblet and P.M. Spira, "A Distributed Algorithm for Minimum Weight Spanning Tree", *ACM Trans. on Program. Lang. and Systems*, Vol. 5, 1983, pp. 66-77.

[GS] O. Goldreich and L. Shrira, "The effects of link failures on computations in asynchronous rings", *Proc. 5th Symp. on Principles of Distributed Computing*, 1986, pp. 174-186.

[GRS] R.L. Graham, B.L. Rothschild and J.H. Spencer, *Ramsey Theory*, John Wiley & Sons, 1980.

[KMZ1] E. Korach, S. Moran and S. Zaks, "Tight Lower and Upper Bounds for some Distributed Algorithms for Complete Network of Processors", *Proc. 3rd Symp. on Principles of Distributed Computing*, 1984, pp. 199-207.

[KMZ2] E. Korach, S. Moran and S. Zaks, "The optimality of distributed constructions of Minimum Weight and Degree Restricted Spanning Trees in a complete network of processors", *Proc. 4th Symp. on Principles of Distributed Computing*, 1985, pp. 277-286.

[LF] N.A. Lynch and M.J. Fischer, "On Describing the Behavior and Implementation of Distributed Systems", *Theoretical Comput. Sci.*, Vol. 13, 1981, pp. 17-43.

[MW] S. Moran and M. Warmuth, "Gap Theorems in Distributed Computing", *Proc. 5th Symp. on Principles of Distributed Computing*, 1986, pp. 131-140.

[MZ] Y. Mansour and S. Zaks, "On the bit complexity of distributed computation in a ring with a leader", *Proc. 5th Symp. on Principles of Distributed Computing*, 1986, pp. 151-160.

[PKR] J. Pachl, E. Korach, and D. Rotem, "A technique for proving lower bounds for distributed maximum-finding algorithms", *Proc. 14th ACM Symp. on Theory of Computing*, 1982, pp. 378-382.

[PS] D. Peleg and A. Schäffer, "Graph Spanners", *J. of Graph Theory*, Vol. 13, 1989, pp. 99-116.

[RK] R. Reischuk and M. Koshors "Lower bound for Synchronous Systems and the Advantage of Local Information" *Proc. 2nd International Workshop on Distributed Algorithms*, Amsterdam, June 1987.