Exact Communication Costs for Consensus and Leader in a Tree

Y. Dinitz
Ben-Gurion University of the Negev, Israel

S. Moran
Technion, Israel

S. Rajsbaum
Compaq CRL, USA

Abstract

We continue the study of communication costs of `Consensus` and `Leader` initiated in [7]. We deal with all scenarios with linear complexity in a tree topology, and prove exact (as opposed to asymptotic) tight bounds for the bit and message complexities. A particular scenario depends on whether the tree size or the size parity is known to the processors. Besides, we show that the message complexity lower bound of [7] for `Consensus` in a ring holds even if the ring size is known to the processors.

Keywords

Distributed Computing, Consensus, Leader Election, Bit Complexity, Message Complexity, Communication Complexity.

1 Introduction

The two most studied tasks in the distributed computing literature are probably `Consensus` and `Leader Election`. These tasks can be viewed as duals in the sense that in `Consensus` all processors have to agree on the same value, while in `Leader` one processor has to output a different value from the rest of the processors. In this sense, `Consensus` is about achieving symmetry, while `Leader` is about breaking symmetry. A large number of results about these tasks exist, including algorithms, lower bounds and applications, in a variety of distributed computing models (see [1, 15]). However, `Consensus` and `Leader` have been considered in different circumstances; typically, `Consensus` is very important in fault-tolerance studies, while `Leader` is the main paradigm for message complexity studies.

In this paper we continue an in-depth study of the relation between the communication costs of `Consensus` and `Leader`, started in our previous paper [7]. The model is the standard asynchronous, failure-free message passing distributed system, with arbitrary but finite link delays and negligible local computation times. There are n links in the network and the processors get different identifiers from the set $Z_M = \{0, \ldots, M-1\}$.

Our interest in the relation between the communication costs of `Consensus` and of `Leader` started by noticing that both tasks have the same asymptotic message complexity in all standard topologies (the lower bound for `Consensus` in rings was proved in [7], the other bounds are well known). In contrast, `Leader` is harder than `Consensus` in a ring in the following sense [7]. Once a leader has been elected, the leader can broadcast a decision value, and thus solve `Consensus` with $O(n)$ messages. On the other hand, even if `Consensus` has been solved, $\Theta(n \log n)$ messages are still needed to solve `Leader`.

The case of a chain turns out to be particularly interesting. The traditional message complexity measure does not distinguish between `Consensus` and `Leader` in a chain: it is easy to see that to solve both, $\Theta(n)$ messages are necessary and sufficient. In [7], a refined measure, *bit complexity*, was introduced, to distinguish between the communication costs of these tasks (see also the relation to communication complexity theory described below). In this paper we show that `Consensus` and `Leader` are not equally difficult in a chain, by proving exact (instead of asymptotic) bounds for message and bit complexities. Moreover, we show how to extend all of our upper bounds to a tree topology.

In this paper we deal with all chain scenarios where `Consensus` and `Leader` have linear complexities. A particular scenario depends on whether the length of the chain, or its parity, is known to the processors, and whether ids are at all processors or at the terminals only. The only scenario that gives raise to a non $O(n)$ complexity (depending also on M), is when the length of the chain is odd and the task is `Leader`. This scenario is where the main asymptotic bit complexity results of [7] were proved; the most interesting case is when ids are given to the two terminal processors only, when the complexity is $\Omega(n \log M)$.

Our results for the tree topology are summarized in Figure 1. The upper bounds for `Consensus` hold for both trees with ids only at the terminals and trees with ids at every processor. For `Leader`, in the case where processors know n, the message lower bound holds even if all processors have ids, while the message upper bound holds even if only the terminals have ids. All our lower bounds are proved for a chain, and thus hold for a general tree (indeed, canceling the additional knowledge on a tree to be a chain can only raise a lower bound).

It is quite surprising that a leader in a tree of an even size can be elected

(that is, an undirected tree can be converted into a rooted directed one) with only $1.5n$ bits, i.e., without sending one bit per direction of every link.

Remark. Our results show that, for tree-like networks, the communication cost of `Consensus` does not depend on the information known on the size (number of processors) of the network. In contrast, the message and, may be, also bit complexity of `Leader` is sensitive to this parameter. In other words, the complexity of breaking symmetry depends on the knowledge that processors have on the network, while that of achieving symmetry does not.

Problem	Knowledge on n	Lower Bound	Upper Bound
`Consensus`	all cases	$2n$ msgs	$2n$ msgs
`Leader`	n known	$n+1$ msgs	$n+1$ msgs
`Leader`	n unknown	$2n$ msgs	$2n$ msgs
`Consensus`	all cases	$2n$ bits	$2n$ bits
`Leader`	unknown even n	$1.5n$ bits	$1.5n$ bits

Figure 1: Our results for the tree topology.

We also extend the `Consensus` message complexity lower bound $\Omega(n \log n)$ for a ring of processors [7] to the case when the processors know n. Specifically, we use the technique of Bodlaender [2]. (This is the only case in this paper where we deal with a topology different from the tree.)

Related Work. Our study of the relation between the communication costs of `Consensus` and `Leader` was started in [7], with the introduction of the bit complexity measure. In particular, we discovered in [7] that the bit complexity of `Consensus` in a chain does not depend on M, while the bit complexity of `Leader` in most cases does depend on M and is sensitive to the information known about the network. Figure 2 presents the results of [7]. It includes the task of electing as the leader the processor with largest identifier, `MaxF`. (The algorithm for `Leader` in the even chain case does not use the ids. In the case of `MaxF`, at the bottom of the figure, the lower bound for a chain holds even under the best possible scheduler.)

Our bit complexity questions are related to *communication complexity* [14]. The basic problem in communication complexity, introduced by Yao [17], asks what is the number of bits that two parties, Alice and Bob, have to communicate to each other in order to compute a function $f(x, y)$ of their respective, private inputs. The model assumes that they send bits one at a time, starting with one or more bits sent by Alice, then some bits sent by Bob, and so on. A large number of results exist on this problem and its variants [12], which consider also more general networks. In particular, the

Problem	Variant	Lower Bound	Upper Bound
`Consensus`	$n = 2$	2	2
`Leader`, `MaxF`	$n = 2$	$2\lfloor \log M \rfloor - 3$	$2\lceil \log M \rceil - 2$
`Consensus`	chain	$\Omega(n)$	$2n$
`Leader`	even chain	$\Omega(n)$	$2n$
`Leader`	odd chain, ids at all	$\Omega(n + \log M)$	$2(n + \log M)$
`Leader`	odd chain, ids at ends	$\Omega(n \log M)$	$2n \log M$
`MaxF`	chain	$\Omega(n \log M)$	$2n \log M$

Figure 2: Previous bit complexity results (parities mentioned are of n).

chain topology has attracted attention. For example, in [5, 16] the relation between the two processor communication complexity of a function, and the communication complexity of the same function on a chain was studied.

From the point of view of communication complexity, our setting introduces a new difficulty to the problem: processors are not distinguished a priori, neither by commonly known identities, nor by a fixed order of sending messages. This is the usual assumption in the distributed computing literature (e.g. see [1, 15]). Moreover, in this literature, it is common that the problem which has to be solved is not necessarily computing a function of the processor inputs, but is specified by a *task*, i.e. an input/output relation (several outputs are allowed for the same input). Also, the output does not have to be the same for all processors.

Organization. This paper is organized as follows. In Section 2 we describe the model of computation, the notion of a scheduler and a particular scheduler used in the rest of the paper, and some definitions about trees. In Section 3 we present our bit and message complexity results for `Consensus`. Section 4 presents the message complexity results for `Leader`, and Section 5 presents the bit complexity results for it. In Section 6 we present the `Consensus` message complexity lower bound for rings when n is unknown. Section 7 contains discussion.

Some descriptions and proofs are shortened in or omitted from this extended abstract because of lack of space.

2 Preliminaries

2.1 Model

In this section we briefly describe the model of computation and the distributed problems we are interested in. (More precise descriptions can be

found in textbooks such as [1, 15].) We consider a standard asynchronous, failure-free message-passing distributed system, consisting of n links connecting pairs of processors. We consider the tree topology; only in Section 6 we consider the ring topology. The processors have distinct identifiers (ids) from the set $Z_M = \{0, \ldots, M-1\}$.

A *distributed algorithm* consists of a set of identical sequential, deterministic algorithms, one for each processor. The algorithms of the processors are identical in the sense that they depend only on the processors identities and number of incident links. A local algorithm includes instructions for sending and receiving messages. All processors are in the same initial state, except for the id and incident link information.

When a processor wakes up spontaneously, or when it receives one or more messages on its incident links, it makes some internal computations according to its local algorithm, and then possibly sends messages on incident links. We assume that the internal computation time is negligible, which implies that messages are sent by a processor only when it wakes up or as an immediate response to receiving a message; we say that the processor is *active* at this moment. Messages are delivered after a finite but otherwise arbitrary delay. Our lower bounds hold even if the messages are delivered in FIFO order, while our algorithms do not require this assumption.

In a *decision task* the processors start with input values and must eventually decide on output values which satisfy the task's specification. In this paper, the inputs are the processors id's, and hence any vector of distinct integers from Z_M, one for each processor, is a possible input to the system. The outputs are binary. When we study the tree topology, we consider also the case where only the terminal processors of the tree have inputs; in particular, for a chain, only the two end processors have inputs.

We are interested in the complexities of the following tasks:

1. `Consensus`: all processors must decide on the same bit. If all the id's are odd they must output 1, if all id's are even they must output 0.
2. `Leader`: one processor decides on 1, the rest decide on 0.

For `Consensus` we refer to the least significant bit of a processor id as to its *input bit*. For `Leader` in a tree, we require also that each non-leader processor knows which of its incident links is directed towards the leader.

We consider the following complexity measures:

- *Message Complexity*: number of messages sent, denoted $MsgC$.
- *Bit Complexity*: number of bits sent, denoted $BitC$.

We consider *worst case* complexities, in which the above figures are maximized over all input configurations and all possible executions.

Remark. Our processors do not have sense of direction [8], an assumption that would be too strong for our problems. For instance, in a chain with sense of direction `Leader` is solved with no communication at all (the leader is the terminal processor on the "left," say), while `Consensus` can be solved with only n bits (the leader broadcasts its bit).

2.2 Schedulers

Given a distributed system, a *scheduler* defines the order in which processors wake up spontaneously and messages are delivered. An *execution* is defined by a given scheduler as follows: initially, all processors are asleep; each next configuration is determined by the current configuration and the decisions made by the scheduler. Notice that upon any decision—waking up a sleeping processor or delivering a message in transit—the corresponding processor executes a local computation (in a negligible time) and may send messages.

We assume that no two messages are delivered to the same processor simultaneously. In this paper we assume that every terminal (leaf) processor of a tree is eventually awaken by the scheduler, unless it has been awaken by receiving a message.

For lower bound proofs for the chain topology, we use the following *backward-preference* scheduler (a variant of the scheduler introduced in [7] by the name "outside-precedence scheduler"). Suppose, a chain terminal, P, has waken up spontaneously. Let us consider the messages sent by it, the processors waken up by these messages, the messages sent from these processors, and so on. Let us call such a process the *wave* generated by P. In each execution there can be two waves—from the two chain terminals—executed simultaneously. The duration of a wave is up to the moment when one of the processors covered by this wave receives a message from a processor in the other wave or when the wave wakes the other terminal; this moment is called the *wave meeting*. Otherwise, a wave can "die" by coming to a *standstill*—when there are no pending messages to be delivered—before the wave meeting. The backward-preference scheduler is defined for a wave from P as follows: it delivers one message at a time, and it is always an undelivered message on the link which is closest to P. This implies that a message on some link e is delivered only when there are no undelivered messages between P and e. Note that the backward-preference scheduler is well defined for two waves executed simultaneously: indeed, before the wave meeting the waves are completely independent.

2.3 Trees

We use the following notation concerning trees. Consider any edge (link) $e = (P, Q)$ in a tree T with $N = n + 1$ nodes. Removal of e divides T into two parts, one of which contains Q. This part of T is the *subtree hanging on e from P*, and Q is its *root*. If for some node Q one of the incident edges, e, is distinguished, then the *subtree rooted at Q* denotes the subtree hanging on e and rooted at Q. We call the number of nodes in a tree or subtree its *weight*. A node P of T is called its *center* if the weight of all subtrees hanging from it is at most $N/2$. It is well known that, in a tree, either there is exactly one center, and then all subtrees hanging on it have the weight strictly less than $N/2$, or there is a *central edge* $e = (P, Q)$ such that both subtrees hanging on e and rooted at P and Q have weight $N/2$, and then there are two centers: P and Q.

3 Consensus

In this section we prove that the complexity of `Consensus` is $2n$. This result applies for all cases considered in this paper: ids everywhere or at the terminals only, n of any parity known or unknown, and complexity in messages or bits.

Our $2n$ algorithm decides on the maximal bit over all processors. Each terminal processor, upon its wake-up, sends its bit to the neighbor. Each intermediate processor, upon receiving the first message, relays the maximum of its bit and the received one (if it has no id, it simply passes the received bit further). When an intermediate processor receives a message after it has sent one or when a terminal is waken by a message, it decides on the maximum of all bits he had ever seen. Then, it relays the decision bit in the reverse direction of the first message that has come to it.

The algorithm for the general tree topology is similar. The main specifics is that each processor waits for the moment of receiving messages over all links incident to it, except for a single one, and then relays the accumulated maximum over this unique remaining link. We extend to such a process the notion of a *wave*. Notice that the unique wave meeting happens either at some link, when two messages have been sent over it in the opposite directions and then such a message is delivered, or when some message reaches the single processor remaining asleep and wakes it (it is always a terminal). The rule of computing the decision bit remains the same, and this bit is relayed in the reverse direction of *all* previously received messages.

It is easy to see that the above algorithm is correct. It sends exactly two messages with a single bit over each link (in opposite directions), i.e., totally $2n$ bits. This proves our first theorem.

Theorem 1 *For a tree,* $BitC(\texttt{Consensus}) \leq 2n$.

The more difficult part is the matching lower bound. Actually, we prove a stronger claim: the lower bound holds for messages (and hence also for bits). We use the backward-preference scheduler (described in Section 2). We assume that inputs are given to the terminal processors only. A similar technique can extend the proof to the case of ids everywhere, although a higher lower bound on M would be needed.

Theorem 2 *For a chain,* $MsgC(\texttt{Consensus}) \geq 2n$, *when* $M \geq 6$.

Proof. We say that a wave *covers* k links, if exactly k consecutive links, started at the wave originator, carry messages in the wave. Henceforth, we name the terminals S and T, and assume S wakes up spontaneously.

For any id N given to terminal S, let us consider the wave from S under the backward precedence scheduler, until the first moment when one of the following events happen:

1. the currently scheduled event is to deliver a message to T, and S has not decided yet (the wave reaches T before S decides),
2. the wave comes to a standstill before reaching T and before S decides, covering k links,
3. S decides, with the wave covering $k < n$ links.

We call the considered part of the execution *the wave caused by* N. Note that giving the same id N to T leads to a symmetrical wave, because the program of T depends only on its input, and all internal processors are identical.

Consider case 1. Clearly, at least n messages were sent up to this moment, in order for the wave to propagate to T. By the backward precedence rules, there are no messages in transit (except for, may be, to T), and S has not decided yet. In order to decide, S must be activated, and since only T is active, at least n messages are needed for this. This results in at least $2n$ messages, as required. Thus, we can assume that there is no id for which case 1 happens.

Assume next that there are at least two ids leading to waves in case 2. Give one of these ids to S and one to T. If the waves from S and T do not meet, we get to a standstill situation where neither S nor T have decided. Therefore, there has to be a moment where the two waves meet, on a link e or in a processor. Consider the first case; the second is similar. At this moment S and T have not decided, and at least $n+1$ messages have been sent: two messages in opposite directions of e, and one message per edge in each wave, allowing them to meet. So far S and T have not decided, at

least $n+1$ messages have been sent, and there are no messages in transit (except for over e). In order for S and T to decide, at least $n-1$ additional messages are needed. Totally, this is $2n$ messages, as required.

Therefore, we can assume that there is no id for case 1, at most one id for case 2, and hence at least 5 ids for case 3. This implies (since the $M > 5$ ids are consecutive) that there are two ids for each parity in case 3, and hence S decides necessarily on its own bit. Indeed, otherwise, giving to T any id with the same bit and considering the backward precedence execution from S, we arrive at an illegal decision of S.

For all waves of the type 3, consider them only until that moment where S decides. Assume first that there are ids of different parities, such that the corresponding waves, where the ids are given to S and T, do not meet. This is impossible because we have just argued that a terminal decides on its own bit. It follows that for every pair of ids of different parities, if we give one to S and one to T, and start executing the corresponding waves, there will be a moment where the waves meet on an edge e (or in a processor, which is proved in a similar way). Since we assume backward precedence scheduler, neither S nor T has decided at this moment. Thus we are in the same situation as above, where $n+1$ messages have been sent for the waves to meet, and in order for S and T to decide, the waves must propagate outward to S and T, sending $n-1$ additional messages. □

4 Leader: Message Complexity

Recall that in the `Leader` problem each processor must know whether it is the leader, and if not, what is the direction from it to the leader. Here we deal with message complexity; bit complexity is considered in the next section.

4.1 Known Size of the Tree

Assume n is known to each processor. We present an algorithm that elects as the leader one of the tree centers. Recall that there is for sure a single center if n is even and may be two centers if n is odd.

Theorem 3 *If n is known then*

$$MsgC(\texttt{Leader}) \leq \begin{cases} n & \text{if } n \text{ is even,} \\ n+1 & \text{if } n \text{ is odd.} \end{cases}$$

Proof. (Outline) Each terminal sends its id and 1 (as its ordinal number from the chain end) to its neighbor. When an intermediate processor receives a message from the nearest chain end side, it passes further the

id and its own ordinal number from this chain end. In case of even n, the unique central processor learns that it is such one, and decides that it is the leader. If n is odd, each central processor receives an id from the other central one; the central processor that has sent the maximal id decides that it is the leader. The direction to the leader for any non-leader processor is the direction that it has sent its single message.

The generalization for the general tree topology uses a technique similar to the one used in [11] for distributed finding the tree centers. Each terminal sends the same as above: its id and 1 (but now 1 means the weight of the subtree rooted at it). Any intermediate processor of degree d waits for the first $d-1$ messages from the roots of subtrees of weight lesser than $n/2$. Then, it either computes that it is not a central one, and then sends the pair (the maximal id received in these messages, the subtree weight) in the direction of the center, or computes that it is the unique center, and then decides that it is the leader, or computes that it is one of the centers, and then sends the (id, weight) pair to the other center and waits for a message from it. □

The following shows that the bound of the previous theorem is tight.

Theorem 4 *If n is known then, for any $M \geq 4$,*

$$MsgC(\texttt{Leader}) \geq \begin{cases} n & \text{if } n \text{ is even} \\ n+1 & \text{if } n \text{ is odd} \end{cases}$$

Proof. We use backward precedence executions. We describe the proof for the case of odd n. The other case is similar.

Assume that there are at least three ids causing the wave to get to a standstill *before* sending any message over the middle link. Then, for two of the ids, the same situation is reached: a leader from the processors participating in the wave generated by the id is or is not chosen. We give these two ids to S and T and execute the waves from both terminals. By the assumption, the waves do not reach each other before the global standstill. Moreover, by our choice, either two leaders are chosen, or no leader is chosen, a contradiction.

Therefore we can assume now that there are at most two ids causing the wave to get to a standstill before sending any message over the middle link. Hence, there are at least two ids such that the wave from them causes sending a message over the middle link. We give the corresponding ids to S and T, and execute the waves so that the first messages over the middle from both sides of the chain are sent simultaneously. Clearly, up to this moment at least $n+1$ messages are sent in the execution, as required. □

4.2 Unknown Size of the Chain

This case also concerns message complexity, but now nothing is known on the value of n. Then, no processor in a chain, upon the first wave reaching it, is able to decide that it is or is not a the tree center. This implies a higher complexity.

Theorem 5 *If n is unknown then $MsgC(\texttt{Leader}) = 2n$. The lower bound holds if $M \geq 3$.*

Proof. (Outline) For the upper bound, we present an algorithm which elects the terminal with the greatest id as a leader. The $2n$ algorithm for a chain is the following trivial one: each terminal broadcasts its id; once a processor learns what the ids of the two terminals are, it elects as leader the one with the larger id.

The generalization to the tree case is also rather simple. The rule of sending messages is as in the algorithm of `Consensus`. However, *every* message contains the maximal id at a terminal in the corresponding subtree. Eventually, each processor learns these maximums from all directions, and decides accordingly.

For the lower bound in a chain when only the terminal processors have ids, we consider executions of the backward-preference scheduler where the terminal T is waken up spontaneously only in the case when the wave from S stops before reaching it. The proof is by contradiction, assuming that there are three ids such that the backward-preference execution from them comes to its end in less than $2n$ messages.

For any such execution, there is at least one link on which at most one message is sent during the entire execution, in the direction from S to T. An important observation: all the processors between the terminal S which starts the algorithm and this link must decide before a message is delivered on this link. We glue two such waves propagating from the two chain terminals, by identifying two such links. It is easy to see that the glued execution is legal; at the moment of sending the messages over the glued link, *all* processors have decided. Now, since there are 3 such ids, we can choose two waves such that either two leaders are elected, one in each wave, or no leader is elected. This gives the desired contradiction. □

5 Leader: Bit Complexity

In this section we consider bit complexity bounds for `Leader` in the case of n even. This improves the results in [7], which presented a simple algorithm with bit complexity $2n$, and a lower bound of $\Omega(n)$ bits. Here we show that the bit complexity is precisely $1.5n$. The case of odd n was treated in [7],

where it was shown that the bit complexity is $\Theta(n \log M)$. We start with the algorithms, which as in [7], do not use ids.

Theorem 6 *If n is known to be even then $BitC(\texttt{Leader}) \leq 1.5n$.*

Proof. (Outline) We describe a $1.5n$ bit algorithm for the tree topology.

- Each terminal processor, when it wakes up, spontaneously or upon receiving a message, decides that it is not the leader and sends 1 to the neighbor.
- Any intermediate processor P computes modulo 2 the bits received; when there remains a single incident link, e, without a message, it flips the computed bit and sends it over e. If this bit is 1, P decides that it is not the leader, but the leader is farther. Moreover, P sends 0 to every processor that has sent to it 0.
- Any intermediate processor, upon receiving 0 from the next processor, decides that it is not the leader, but the leader is farther.
- Any intermediate processor, upon receiving 1 from the next processor, decides that it is the leader.

To see that the algorithm is correct, observe that the bit sent farther from any processor P is exactly the parity of the number of processors in the subtree rooted at P. Thus, the unique link on which the waves meet carries 0 in one direction and 1 in the other. The rest of the proof is not hard.

In the case of a chain, the numbers of bits sent over links are alternating 1 and 2, from any chain end to the wave meeting link; hence $1.5n$ bits totally. In the case of a tree, there are also alternating loads 1 and 2 from the tree leaves. Since 1's begin, the total number of bits can be shown to be at most $1.5n$. □

Theorem 7 *If n known to be even then $BitC(\texttt{Leader}) \geq 1.5n$, when $M \geq 5$.*

Proof. The proof is a refining of the proof technique of Theorem 5. Again, we suppose for simplicity that only terminals have ids. Assuming, to the contrary, that there is an algorithm with better complexity, we choose some even chain length n_0 such that the algorithm is supposed to use strictly less than $1.5n_0$ messages in any execution in any chain of length n_0.

We use the backward-preference scheduler as defined in the proof of the lower bound for `Consensus`. Let us consider backward-preference executions from all ids in the chain of length n_0. Assuming there are at least 5 ids, one of cases 1 and 2 below must hold:

Case 1: There are three ids such that the wave from them stops before waking up the second terminal. Then, there are either two ids such that elect a leader in the area covered by this wave, or two ids such that it is supposed that the leader is outside this area. Let us consider a $2n_0$ length chain with these two ids given to the terminals. The execution under the backward-preference scheduler from each side stops without wave meeting. Two leaders elected is a contradiction, and also no leader elected is a contradiction.
Case 2: If case 1 does not hold, there are three ids such that in the backward-preference execution from them each edge carries at least one message. It is easy to see that for any such execution, there are two *consecutive* links with exactly one message sent on each during the entire execution. Moreover, all the processors between the originator of the execution and such a link pair must decide before sending the single message on any such link, since they will never be activated again in this execution.

Let us glue two such waves by identifying some two links in some pairs as above, so that the length of the resulted chain is even (it is always possible even if we fix one of the links in one of these pairs, since we can select any of the two consecutive links). Also, we glue the executions so that the (at most) two messages on the identified link are sent simultaneously. It is easy to see that the glued execution is legal. As observed above, at the moment of sending the messages over the glued link, *all* processors have decided. Now, choosing two waves where a leader was elected inside the covered area or two waves where no leader was elected, we arrive at a contradiction. □

6 Consensus in a Ring

In this section, we improve the message complexity lower bound of [7] to the case where n is known:

Theorem 8 *For a ring of n processors that know n, $MsgC(\texttt{Consensus}) = \Omega(n \log n)$.*

The proof is based on the technique in [2], and it holds even when the size of the ring, n, is known. Consider any `Consensus` algorithm A for a ring of size n. Let I be a set of identities. We say that A is *active on I* if the following holds: In any synchronous execution of A on ring in which all the identities are taken from I, and for any time $t < n/4$, at least one message is sent by some processor in time t. The following theorem is implicit in [2]:

Theorem 9 ([2], Lemma 5.1, Theorem 5.8) *Let $|I| \geq cn$, for $c > 1$. Then the average message complexity of any protocol which is active on I, where the average is taken over the worst case complexity over all rings with identities taken from I, is $\Omega(n \log n)$.*

Let I_0 be the set of all even id's, and I_1 be the set of all odd id's. Theorem 8 follows from Theorem 9 above and the following Lemma:

Lemma 1 *For any consensus algorithm A, A is active either on I_0 or on I_1.*

Proof. (Outline) Otherwise, there exist two rings - R_0 with inputs from I_0 and R_1 with inputs from I_1, in which all the processors are quiescent at time $n/4-1$. Then at time $n/4-1$ all processors in R_1 decide on 1, and all processors in R_0 decide on 0. Consider now a ring constructed by cutting and pasting half of the ring R_1 to half of R_2. Let P (Q resp.) be a processor in R which is surrounded by chains of $n/4-1$ processors from I_0 (I_1 resp.)on both sides. Since in a synchronous execution of less than $n/4$ P (Q resp.) cannot distinguish between R and R_0 (R_1 resp.), in a synchronous execution of A on R, P will decide on 0 and Q will decide on 1 - a contradiction. □

Remark. An argument similar to the one in Lemma 1 can be used to prove that $\Omega(n \log n)$ messages are necessary to solve `Consensus` in a complete graph. This extends the `Leader` lower bound of [13] for complete graphs.

7 Discussion

1. Our results show that when nothing is known about n, `Leader` is at least as hard as (and sometimes harder than) `Consensus`, both in terms of messages and bits; `Leader` is easier than `Consensus` when some information about n, which can be used for symmetry breaking, is known.

2. The decision of distinguishing terminal processors both as possessing ids and as being able to wake-up spontaneously, seems natural, from the practical, as well as from the theoretical point of view. Suppose, for example, that the terminals are important processors (e.g., users), while the other ones are interconnection stations which only relay messages (see also [6] for such a point of view). Also, in communication complexity studies [5, 16] Alice and Bob, the chain terminals, are the only processors with inputs. Further work is needed to investigate other alternatives.

3. If n is known to be even, we have shown that $1.5n$ bits are necessary and sufficient to solve `Leader` in a chain. It is an open question whether this bound holds if n is even and its actual value is known.

4. All our lower bounds for the tree topology are proved for a chain. It may be of interest whether these bounds can be proved for other special tree topologies.

References

[1] Hagit Attiya and Jennifer Welch, *Distributed Computing: Fundamentals, Simulations and Advanced Topics,* McGraw–Hill, England, 1998.

[2] Hans L. Bodlaender, "New lower bound techniques for distributed leader finding and other problems on rings of processors," *Theor. Comput. Sci.* **81** (1991), 237–256.

[3] Hans L. Bodlaender, Shlomo Moran, Manfred K. Warmuth, "The distributed bit complexity of the ring: From the anonymous to the non-anonymous case", *Information and Computation* **114** (2), pp. 34–50, 1994.

[4] James E. Burns, "A formal model for message passing systems," Technical Report TR-91, Computer Science Dept., Indiana University, Bloomington, September 1980.

[5] Martin Dietzfelbinger, "The linear-array problem in communication complexity resolved," in *Proceedings of the 29th ACM Symposium on Theory of Computing*, pp. 373–382, 1997.

[6] Ye. Dinitz, T. Eilam, Sh. Moran, and Sh. Zaks. "On the Total$_k$-Diameter of Connection Networks," In print in *Theoretical Computer Science.*

[7] Yefim Dinitz, Shlomo Moran, Sergio Rajsbaum, "Bit Complexity of Breaking and Achieving Symmetry in Chains and Rings," in *Proc. of the 31st ACM Symp. on the Theory of Computation,* pages 265–274, 1999.

[8] P. Flocchini, B. Mans, N. Santoro, "Sense of direction: formal definition and properties," In Proc. of 1st Colloquium on Structural Information and Communication Complexity (Sirocco), Ottawa, Canada, 1994, pp. 9–34.

[9] Robert G. Gallager, Pier A. Humblet, Philip M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Trans. Prog. Lang. Syst.,* **5**(1):66–77, Jan. 1983.

[10] Daniel S. Hirschberg, J.B. Sinclair, "Decentralized extrema-finding in circular configurations of processes," *Commun. ACM* **23** (11), pp. 627–628, 1980.

[11] Ephraim Korach, D. Rotem and N. Santoro, "Distributed algorithms for finding centers and medians in networks," *ACM Trans. Program. Lang. Syst.*, **6** (3), pp. 380–401, 1984.

[12] Eyal Kushilevitz, Nathan Lineal, Rafail Ostrovsky, "The linear-array conjecture in communication complexity is false," *Proc. of the 28th Annual ACM Symp. Th. Comp.* (1996), pp. 1–10.

[13] Ephraim Korach, Shlomo Moran, Shmuel Zaks, "Tight upper and lower bounds for some distributed algorithms for a complete network of processors," *Proc. of the 3th Annual ACM Symp. Princ. Dist. Comp.* (1984), pp. 199–207.

[14] Eyal Kushilevitz, Noam Nisan, *Communication Complexity*, Cambridge University Press, 1997.

[15] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, Inc. 1996.

[16] Prasoon Tiwari, "Lower bounds on communication complexity in distributed computer networks," *J. Assoc. Comput. Mach.* **34** (1987), 921–938.

[17] Andrew C. Yao, "Some complexity questions related to distributed computing," *Proc. of the 11th ACM Symp. on Theory of Comp.* (1979), pp. 209–213.

Yefim Dinitz is with the Department of Computer Science, Ben-Gurion University of the Negev, P.O. Box 653, Beer-Sheva 84105 Israel. Email: dinitz@cs.bgu.ac.il

Shlomo Moran is with the Department of Computer Science, The Technion, Haifa, 32000 Israel. This work has been supported in part by the Bernard Elkin Chair for Computer Science. Email: moran@cs.technion.ac.il

Sergio Rajsbaum is with the Cambridge Research Lab of Compaq, One Kendall Square Build. 700, Cambridge, MA 02139. On leave from the Instituto de Matemáticas, UNAM, Ciudad Universitaria, D.F. 04510, Mexico. Email: rajsbaum@{ crl.dec.com,math.unam.mx}. Part of this work was done while visiting the Technion.