Two Lower Bounds in Asynchronous Distributed Computation

PAVOL DURIS
Slovak Academy of Sciences

AND

ZVI GALIL*
Columbia University and Tel-Aviv University

Received February 8, 1988; revised January 30, 1989

We introduce new techniques for deriving lower bounds on message complexity in asynchronous distributed computation. These techniques combine the choice of specific patterns of communication delays and crossing-sequence arguments with consideration of the speed of propagation of messages, together with careful counting of messages in different parts of the network. They enable us to prove the following results, settling two open problems:

- An $\Omega(n \log^* n)$ lower bound for the number of messages sent by an asynchronous algorithm for computing any nonconstant function on a bidirectional ring of $n$ anonymous processors.

- An $\Omega(n \log n)$ lower bound for the average number of messages, sent by any maximum-finding algorithm on a ring of $n$ processors, in case $n$ is known.

INTRODUCTION

Consider the following model. We have a bidirectional asynchronous anonymous ring of $n$ processors ([1, 6]). There is no leader among the processors. All processors run the same program, which may depend on the size of the ring. All processors compute the same function $f: \Sigma^n \rightarrow \{0, 1\}$, where $\Sigma$ is an arbitrary finite alphabet. The input of each processor is a letter of $\Sigma$, and the processors compute $f(x)$, where $x$ is the concatenation of the $n$ inputs beginning with any processor on the ring. We assume that for every input $x \in \Sigma^n$ and for any possible pattern of communication delays (or scheduling of the messages sent) all the processors eventually stop. Upon termination all processors are in one of two states: either they all accept (which corresponds to $f(x) = 1$) or they all reject (which corresponds to $f(x) = 0$).

* The work of Zvi Galil was supported in part by NSF Grants DCR-85-11713 and CCR-86-05353.
In [1] Attiya et al. considered the following function: \( f(x) = 1 \) if \( x \) is a cyclic shift of a string in \( 0(01)^* \) and is 0 otherwise. They showed that if \( n \), the size of the ring, is assumed to be odd, then the function can be computed in \( O(n) \) messages. Similar nonconstant functions computable in \( O(n) \) messages can be defined if the size of the ring is assumed to have any fixed constant nondivisor. They left as an open problem whether a similar result can be obtained without restrictions on the size of the ring. In [6] Moran and Warmuth defined a nonconstant function and proved that its message complexity is at most \( O(n \log^* n) \). Our main result answers the open problem in [1] in the negative and shows that the upper bound in [6] is best possible:

**Theorem 1.** If \( f : \Sigma^n \to \{0, 1\} \) is a nonconstant function, then any asynchronous algorithm for computing \( f \) on a bidirectional ring of \( n \) anonymous processors requires at least \( cn \log^* n \) messages in the worst case for some constant \( c > 0 \).

In [6], Moran and Warmuth showed that any nonconstant function requires \( \Omega(n \log n) \) bits on an anonymous ring of size \( n \), while it is easy to construct nonconstant functions with \( O(n \log n) \) bit complexity on such a ring. They refer to this phenomenon as a gap in complexity between constant and nonconstant functions. (When the function is constant the bit complexity is 0.) They also left open the question whether a gap exists when we consider the message complexity. Their lower-bound techniques were not sufficient for establishing the gap which Theorem 1 exhibits. Theorem 1 deals with the more general setting that allows general messages. Moreover the result in [6] did not exclude the possibility of \( O(n) \) message complexity of nonconstant functions. For example, the algorithm of [1] mentioned above has \( O(n) \) message complexity but \( \Theta(n \log n) \) bit complexity.

In Section 1 we prove Theorem 1. Our arguments consider the speed of propagation of certain messages as well as crossing sequences (i.e., cut and paste) and specific choices of communication delays to fool the algorithm and derive a contradiction.

Our second main result concerns the problem of maximum-finding on a ring of processors, which is one of the basic problems in distributed computation. Its solutions are used as building blocks in more complicated algorithms. It has been studied quite extensively. We consider a ring of \( n \) processors \( p_1, p_2, \ldots, p_n \), and let \( L = \{s_1, s_2, \ldots, s_m\} \) be a set of labels (distinct integers). Assume that for \( i = 1, \ldots, n \), \( p_i \) is labeled by \( r_i \in L \) and every two processors are labeled by distinct labels. We consider asynchronous message-driven algorithms in which all processors start simultaneously, the communication channels are first-in first-out, and all processors eventually stop after computing the maximum label (see [7]).

There are two different versions of the problem, depending on whether or not \( n \) is known to the processors. Also, one can consider the worst-case message complexity or the average message complexity. In the latter case we average the message complexity over all possible distinct label assignments to the \( n \) processors. For each such assignment we consider the worst pattern of communication delays.
Consequently, we have four cases to consider. There is yet another distinction between unidirectional and bidirectional rings, but similar results were obtained for both subcases: usually lower bounds are proved in the bidirectional case and upper bounds for the unidirectional case. This distinction is only important for determining the best constants in the bounds.

$O(n \log n)$ upper bounds for all four cases have been known for some time (see for example [5, 8]). Burns [4] and Pachl et al. [7] proved $\Omega(n \log n)$ lower bounds in all cases but one. Their techniques did not suffice for determining the average message complexity in case $n$ is known, and no nontrivial (nonlinear) lower bound was known. Bodlaender [2] proved an $\Omega(n \log n)$ lower bound for unidirectional algorithms which use only comparisons between labels. Our second result almost completes the picture:

**Theorem 2.** Let $n$ be a power of 2. If the label set $L$ is sufficiently large, then any maximum-finding algorithm for a bidirectional ring of size $n$ labeled by $L$ in which the processors know $n$ has average message complexity at least $cn \log n$ for some constant $c > 0$.

In Section 2 we prove Theorem 2. We choose two types of specific communication delays: the first leads to contradiction by forcing the algorithm to terminate without the correct answer, and the second forces the algorithm to send many messages needed for the desired lower bound. The proofs also use arguments that consider the speed of propagation of certain messages, as well as a special way of counting the messages in different parts of the ring. It would be interesting to prove a similar theorem for a general $n$ (not necessarily a power of 2).

1. **The Proof of Theorem 1**

We start with a sketch of the proof. We first fix an arbitrary algorithm $A$ computing a function $f$. Then we fix a particular scheduling of messages in which transmission takes one unit of time and computation takes no time. Thus, we deal with a synchronous computation and we can speak about time. We then define a crossing sequence at a link.

In Lemma 1 we observe two simple facts: (1) The crossing sequence of length $l$ at a link depends only on the input to processors at distance at most $l$ from this link; and (2) a message at a link at time $T$ must be preceded by a message at that link or one of the neighboring links at time $T - 1$. Applying (2) repeatedly implies that a message at a link which is in the middle of a segment of $2l$ processors at time $T \geq l$ must be preceded by a message in this segment at each of the previous $l$ time steps.

In Lemma 2 we find an input string $z$ such that in the computation of $A$ on $z$ no processor accepts or rejects before time $n/4$. We fix the input to be $z$ and then establish the lower bound by estimating the number of messages in the computation of $A$ on $z$. 
We consider roughly $\log^* n$ disjoint time intervals, each exponentially longer than the previous one. Lemma 3 shows that at any time interval in any segment of length proportional to the length of the time interval the number of messages sent must be at least linear in this length. Consequently, at least a linear number of messages must be sent in each time interval, which completes the proof of Theorem 1.

For convenience we denote the anonymous processors by $p_1, p_2, \ldots, p_n$. (Professor $p_i$ does not know $i$.) Without loss of generality we assume that the ring $R$ of size $n = m$! is oriented; i.e., all processors in it agree on the same “right” and “left” directions. Consider an algorithm $A$ which computes a nonconstant function $f$ on the ring $R$ (where both are arbitrary but fixed). For input $w = w_1 w_2 \cdots w_n$ we choose a particular pattern of communication delays: All processors start at time zero, internal computations take no time and links are “synchronized”, i.e., it takes one unit of time to traverse the link. Therefore we can speak of time $0, 1, 2, \ldots$ in the computation of $A$ on $w$. Recall that $n = m!$ and assume first that $m \geq 192$ and $\Sigma = \{0, 1\}$.

In the proof we use the notion of a “segment” of the ring $R$ and the notion of a “crossing sequence” at a link of $R$. For $1 \leq i < j \leq n$, the segment $[i, j]$ consists of the processors $p_i, p_{i+1}, \ldots, p_j$ and all the links associated with them including the first (leftmost) and last (rightmost) links. A segment is a segment $[i, j]$ for some $1 \leq i < j \leq n$. The length of a segment $s = [i, j]$ is $|s| = j - i + 1$ and the input to $s$ is the string $w_i w_{i+1} \cdots w_j$.

Let $M$ be the set of all possible messages and let $\varepsilon \notin M$ denote the empty message (not counted in the message count). The crossing sequence at a link $k$ until time $T \geq 0$ is a pair $(r, l)$, $r, l \in (M \cup \{\varepsilon\})^{T+1}$, where for $0 \leq t \leq T$, $r_t, l_t$ is the message sent right (left) on $b$ at time $t$. If $r_t = l_t = \varepsilon$ we say that the link $b$ is passive at time $t$ in the computation of $A$ on $w$. Otherwise we say that $b$ is active. Note that in every time unit until every processor halts at least one link is active.

The configuration of a processor $p$ at time $t \geq 0$ in the computation of $A$ on $w$ is the triple $(q, m_L, m_R)$, where $q$ is the state of the processor $p$ (running $A$) immediately after performing its internal computation, and $m_L$ ($m_R$) is the message (if there is any) sent by $p$ to the left (to the right) at time $t$ in the computation of $A$ on $w$; if there is no such message then $m_L$ ($m_R$) is $\varepsilon$. The configuration of a segment $[i, j]$ at time $t \geq 0$ in the computation of $A$ on $w$ is the $(j - i + 1)$ tuple $(a_i, a_{i+1}, \ldots, a_j)$, where $a_i$ is the configuration of $p_i$ at time $i$ in the computation of $A$ on $w$ for $i = i, i + 1, \ldots, j$.

The proof of Theorem 1 requires some lemmas. Lemma 1 follows by considering the propagation speed of messages.

**Lemma 1.**

(a) Let $i, T, T'$ be positive integers such that $0 \leq i - T, i + T \leq n$, and $T < T'$. If the link connecting $p_i$ and $p_{i+1}$ is passive at every time $t = T, T + 1, \ldots, T'$ in the computation of $A$ on $w$, then the crossing sequence at this link until time $T'$ is uniquely determined by the string $w_{i-T+1} w_{i-T+2} \cdots w_{i+T}$ and by the algorithm $A$. 


(b) Let $1 \leq i < j \leq n$ and $i + 1 < j - 1$. If all processors of the segment $[i, j]$ are passive at time $t$ in the computation of $A$ on $w$, then all processors of the segment $[i + 1, j - 1]$ are passive at time $t + 1$.

Proof. (a) The proof is obvious for $T = 1$. Assume $T \geq 2$. First, observe that the configuration of the segment $[i - T + 1, i + T - t - 1]$ at time $t + 1$ in the computation of $A$ on $w$ is uniquely determined by the configuration of the segment $[i - T + 1, i + T - t]$ at time $t$ and by $A$ for every $t = 0, 1, 2, \ldots, T - 2$. Clearly, all these configurations are determined by the configuration of $[i - T + 1, i + T]$ at time $t = 0$, i.e., by the string $w_{i - T + 1} \cdots w_{i + T}$ and by algorithm $A$. On the other hand, they uniquely determine the crossing sequence mentioned above until time $T - 1$. To complete the proof, note that the link is passive after time $T - 1$ and until time $T$.

(b) The proof follows from the fact that if three consecutive links are passive at some time in the computation of $A$ of $w$, then the middle link is passive at the next time in the computation of $A$ on $w$.

Corollary 1. Let $1 \leq i < n$. If the link connecting $p_i$ and $p_{i+1}$ is active at time $t$, then at least one link of the segment $[i - l, i + l + 1]$ is active at time $t - l$ for every $l = 0, 1, 2, \ldots, \min\{t, i - 1, n - i - 1\}$.

Proof. Suppose that Corollary 1 is false and apply part (b) of Lemma 1 to derive a contradiction.

Corollary 2. Let $1 \leq j_1 < i < j_2 \leq n$. If the link connecting $p_i$ and $p_{i+1}$ is active at time $t$, then at least one link of the segment $[j_1, j_2]$ is active at time $t - l$ for every $l = 0, 1, 2, \ldots, \min\{t, i - j_1, j_2 - i - 1\}$.

Proof. Observe that all links of the segment $[i - l, i + l + 1]$ are some links of the segment $[j_1, j_2]$ for every $l = 0, 1, 2, \ldots, \min\{t, i - j_1, j_2 - i - 1\}$ and apply Corollary 1.

Lemma 2. There is an input $z \in \{0, 1\}^n$ such that no processor of the ring accepts or rejects before time $n/4$ in the computation of $A$ on $z$.

Proof. We use the method of [1] or [6]. Consider the computation of $A$ on input $0^n$. The input is completely symmetric. All processors run the same algorithm and thus are in the same state of the algorithm at any given time. At least one message is sent by each processor at each time until some time $T$ at which no message is sent. Thus all the processors terminate at time $T$ after sending at least $nT$ messages altogether. If $T \geq n/4$, then the message complexity is at least $n^2/4$, which is much more than we need for completing the proof of Theorem 1. So assume $T \leq n/4 - 1$. Since the function $f$ is nonconstant, there is an input $z$ in $\{0, 1\}^n$ such that $f(z) \neq f(0^n)$. We show that $z$ has the desired property. Assume for contradiction that there is a processor $p_j$ which terminates at time $T' \leq n/4 - 1$ in
the computation of $A$ on $z$. Since the ring is invariant under circular shifts we may assume that $j = T' + 1$. Now consider the computation of $A$ on input $z_1 0^{2T-1}$, where $z_1$ is the prefix of $z$ of length $n - 2T - 1 > n/2 > 2T' + 1$. In this computation the processor $p_i$ terminates with the result $f(z)$ but the processor $p_{n-T}$ terminates with the result $f(0^n)$, which is a contradiction.

The next lemma requires the following definitions:

$$d_i = 4, \quad d_{i+1} = 3d_i 2^d, \quad \text{for} \quad i \geq 1$$

$$T_i = d_i/4, \quad \text{for} \quad i \geq 1.$$  

Let $k$ be the integer such that

$$d_k \leq m < d_{k+1}.$$  

Recall that $n = m!$ and $m \geq 192$. Hence $d_2 = 192 \leq m$ and $k \geq 2$. Theorem 1 follows from the following lemma.

**Lemma 3.** Let $1 \leq i \leq k - 1$, let $S$ be an arbitrary segment of length $d_{i+1}$, and let $z$ be the string of Lemma 2. Then there are at least $|S|/12$ messages sent in internal links of $S$ during the time period that starts at $t = T_i$ and ends at $t = T_{i+1} - 1$ in the computation of $A$ on $z$.

We first sketch the proof of Lemma 3. We consider the segment $S$ as a sequence of smaller segments and distinguish between two cases. In the first case, for many of the small segments a message is sent in the middle link at a certain late part of the corresponding time interval. In this case, Lemma 1 identifies linear number of messages sent in each of these small intervals and altogether a linear number of messages in $S$ in this time interval. In the second case, most of the middle links are passive during the late part of the intervals. We then claim that one of the processors of $S$ located near the middle of $S$ must send a message at the end of the time interval. Lemma 3 follows from the claim by Lemma 1. The claim is established by a quite elaborate cut and paste technique.

**Proof of Lemma 3.** By (1) the segment $S$ consists of $b = 3 \cdot 2^d$ consecutive segments of length $d_i$ which we denote by $s_1, s_2, \ldots, s_h$. We say that a segment $s_j$ is rich if there is a time $t$, $2T_i \leq t < T_{i+1}$, such that the middle link of $s_j$ was active at time $t$, and we say that $s_j$ is poor otherwise. Let $g = 2^d + 1$ and $h = g + 2^d + 1 - 1$ and consider the segments $s_g, s_{g+1}, \ldots, s_h$ (the middle 2 of the 3 segments) and distinguish between two cases: in Case 1 at least $2^d$ of them are rich, and in Case 2 more than $2^d$ of them are poor.

**Case 1.** There are at least $2^d$ rich segments among the segments $s_g, s_{g+1}, \ldots, s_h$. Let $s_j$ be an arbitrary rich segment. By (2), $|s_j| = d_i = 4T_i$. Thus, by Corollary 2, there is at least one message in the segment $s_j$ at time $t - l$ in the computation of $A$ on $z$ for every $l = 0, 1, 2, \ldots, T_i - 1$. Note that $T_{i+1} - 1 \geq t - l \geq T_i$, by the
definition of rich segment. Summing up these messages over all rich segments among the segments $s_g, s_{g+1}, \ldots, s_h$, we obtain the desired number of messages.

**Case 2.** There are at least $2^d_i + 1$ poor segments among the segments $s_g, s_{g+1}, \ldots, s_h$. Since there are $2^d_i$ strings over $\{0, 1\}$ of length $d_i$, we have that there are two indices $p, q$ ($g < p < q < h$) and there is a string $u$ over $\{0, 1\}$ of length $d_i$ such that the segments $s_p$ and $s_q$ are poor, and $u$ is the input to both segments $s_p$ and $s_q$ in the computation of $A$ on $z$. In order to complete Case 2 we now prove the following claim.

**Claim 1.** At least one link among all links of the segments $s_g, s_{g+1}, \ldots, s_h$ is active at time $T_i+1 - 1$ in the computation of $A$ on $z$.

**Proof.** Assume for contradiction that all these links are passive at time $T_i+1 - 1$ in the computation of $A$ on $z$. By $v$ we denote the string over $\{0, 1\}$ of length $(q - p - 1) d_i$ which is the input to the segment formed by the segments $s_{p+1}, s_{p+2}, \ldots, s_{q-1}$ in the computation of $A$ on $z$. Note that $z = xuwy$ for some $x$ and $y$. Let $u_1$ be the left half and $u_2$ the right half of $u$. Let $s'_p$ be the right half of the segment $s_p$ and let $s'_q$ be the left half of the segment $s_q$. Let $s$ be the segment of length $(q - p) d_i$ formed by the segments $s'_p, s_{p+1}, \ldots, s_{q-1}, s'_q$. Note that the string $u_2 vu_1$ is the input to the segment $s$ in the computation of $A$ on $z$. For $0 \leq t \leq T_i+1 - 1$ we denote by $c_t$ the configuration of the segment $s$ at time $t$ in the computation of $A$ on $z$.

We use below two simple observations. The first one is that the crossing sequences at the middle link of $s_p$ and at the middle link of $s_q$ until time $T_i+1 - 1$ in the computation of $A$ on $z$ are the same. To prove it use the facts that $s_p$ and $s_q$ are poor (their middle links are passive at $t$, $2T_i \leq t \leq T_i+1 - 1$), and the string $u$ is the input to $s_p$ and to $s_q$ in the computation of $A$ on $z$, and apply (a) of Lemma 1 twice with $T = 2T_i$, $T' = T_i+1 - 1$, and the two middle links. The second observation is obvious: The configuration of an arbitrary segment at time $t + 1$ is uniquely determined by the configuration of this segment at time $t$, by the message received (from the left) by the leftmost processor of this segment at time $t$, by the message received (from the right) by the rightmost processor of this segment at time $t$, by the algorithm $A$.

Now consider the computation of $A$ on the input $z' \equiv (u_2 vu_1)^{n/|u_2 vu_1|}$. Note that $|u_2 vu_1|$ divides $n = m!$, since $|u_2 vu_1| = |s| \leq |S| = d_i+1 \leq d_k \leq m$, by (3). Divide the whole ring into $n/|u_2 vu_1|$ consecutive segments of length $|u_1 vu_2|$, such that the string $u_2 vu_1$ is the input to every one of them in the computation of $A$ on $z'$. One can show, by induction on $t$ and by the two observations above, that every such segment is in the same configuration $c_t$ (introduced above) at time $t$ in the computation of $A$ on $z'$ for every $0, 1, 2, \ldots, T_i+1 - 1$. In particular, all these segments are in the configuration $c_{T_i+1 - 1}$ at time $T_i+1 - 1$ in the computation of $A$ on $z'$. Recall that the segment $s$ is in the configuration $c_{T_i+1 - 1}$ at time $T_i+1 - 1$ in the computation of $A$ on $z$. Since $T_i+1 - 1 < T_k = d_k/4 \leq m/4 \leq n/4$, there is neither an accepting nor a rejecting state in the configuration $c_{T_i+1 - 1}$, by Lemma 2. By the
assumption at the beginning of the proof of the claim and by the definition of \( s \), all the links of \( s \) are passive at time \( T_{i+1} - 1 \) in the computation of \( A \) on \( z \); i.e., there is no message in \( c_{T_{i+1} - 1} \). Consequently, all links of the whole ring are passive at time \( T_{i+1} - 1 \) in the computation of \( A \) on \( z' \). Thus, the computation of \( A \) on \( z' \) has stopped before time \( T_{i+1} \). Moreover, since the configuration \( c_{T_{i+1} - 1} \) contains neither an accepting nor a rejecting state, this computation has stopped without any output—a contradiction, which completes the proof of the claim. 

In order to complete Case 2 (and the proof of Lemma 3), observe, using the definitions of the segment \( S \) and the numbers \( g, h \), that the active link mentioned in the claim connects two processors \( p_j \) and \( p_{j+1} \) such that all links (and all processors) of the segment \([j - d_{i+1}/6 + 1, j + d_{i+1}/6]\) are some links (some processors) of the segment \( S \), since \( d_{i+1}/6 < j < 5d_{i+1}/6 \). Consequently, by Corollary 2, there is at least one message in the segment \( S \) at time \( T_{i+1} - 1 - l \) for every \( l = 0, 1, 2, \ldots, d_{i+1}/6 - 1 \). Note that \( T_{i+1} - 1 > T_{i+1} - 1 - l > T_i \) by (1) and (2). This completes the proof of Lemma 3. 

**Theorem 1.** Let \( n = m! \), \( m > 1 \). If \( f: \{0, 1\}^n \rightarrow \{0, 1\} \) is a nonconstant function, then any asynchronous algorithm for computing \( f \) on a bidirectional ring of \( n \) anonymous processors requires at least \( \frac{1}{2} n (\log^* n - 6) \) messages in the worst case.

**Proof.** If \( m < 192 \), the lower bound is immediate since the number of messages is at least \( n/4 \) by Lemma 2. So assume \( m \geq 192 \). By Lemma 3, the total number of all messages in the computation of \( A \) on \( z \) is at least \((k - 1) n/12 \). We now estimate \( k \). Let \( e_1 = 4 \), \( e_{i+1} = 2^{2^i} \), for each \( i \geq 1 \). Thus, by (1), \( d_i \leq e_i \) for each \( i \geq 1 \). Now, \( \log^* e_i = 2i \) for each \( i \geq 1 \). These facts with (3) yield

\[
\log^* n = \log^* (m!) \leq \log^* m + 2 \leq \log^* d_{k+1} + 2 \leq \log^* e_{k+1} + 2 = 2k + 4.
\]

Theorem 1 follows from this estimate. 

The only place we used the assumption that \( |\Sigma| = 2 \) was in the proof of Lemma 3. The proof can be easily modified to handle any finite size alphabet resulting in a different constant in Theorem 1.

**2. The Proof of Theorem 2**

For proving Theorem 2, we fix the algorithm \( A \) and consider a different scheduling for the processors. Then we define certain sets of strings and prove some of their properties.

In this section we also allow segments \([i, j]\) with \( i = j \) as well as those with \( i > j \) which "go around the ring." A segment is waiting if all messages in all its interior links have arrived and no processor in it is able to send any message before receiving a message in the first or last link.
We fix a constant $\alpha$, $0 < \alpha < \frac{1}{2}$. We consider $A$, an arbitrary maximum-finding algorithm on a ring of size $n$, where $n \geq 8$ is power of two, and with a label set $L$ that contains $m$ distinct labels. We choose $m$ large enough so that the following inequality holds for $l = 2, 4, 8, ..., n/4$:

$$(n-1)(n+1) \sum_{k=1}^{n-l} \binom{m-l}{k-l} < (1-2\alpha)^{n/l} \frac{(m(m-1) \cdots (m-2l+1))^{n/l}}{(m(m-1) \cdots (m-l+1))^{n/l+1}}. \tag{4}$$

This is possible since if we fix $n$, then the left-hand side of (4) grows like $m^{n-1/l}$, while the right-hand side of (4) grows like $m^{n-l}$.

For a string $r = r_1r_2 \cdots r_l$, $r_i \in L$ for $1 \leq i \leq l$, we denote by $\text{set}(r)$ the set of all the different $r_i$'s in $r$. For $l = 1, 2, 4, 8, ..., n$ we define $S_l = \{r \mid r = r_1r_2 \cdots r_l, r_i \in L$ for $1 \leq i \leq l$, and $r_i \neq r_j$ for $i \neq j\}$.

For $l = 1, 2, 4, 8, ..., n$ and for each $r \in S_l$ we denote by $c(r)$ the following computation of $A$ in the segment labeled by $r$. If $r \in S_1$, then $c(r)$ is "do nothing." If $r = rs' \in S_{l/2}$, $s, s' \in S_l$, then $c(r)$ is defined by keeping the first (leftmost), last (rightmost), and middle links of the segment labeled by $r$ very slow, executing $c(s)$ and $c(s')$. Then we increase the speed of the middle link and continue the execution of $A$ in an arbitrary (but fixed) way till the segment labeled by $r$ is waiting. Let $|c(r)|$ be the number of messages sent during $c(r)$. In the proof below we use the following sets for $l = 2, 4, 8, ..., n/4$:

$$C_{2l} = \{ss' \mid s, s' \in S_{2l}, s_i s_{i+1} \in H_{2l} \text{ for } 1 \leq i \leq h-1\}$$

and

$$H_{2l} = \{ss' \mid s, s' \in S_{2l}, ss' \notin C_{2l}, s's \notin C_{2l}\}.$$

**Lemma 4.** For $l \in \{2, 4, 8, \ldots, n/4\}$ and any fixed integer $h \geq 2$, (a) and (b) hold for the sets

$$V = \{s_1s_2 \cdots s_h \mid s_i \in S_l \text{ for } 1 \leq i \leq h, s_i s_{i+1} \in H_{2l} \text{ for } 1 \leq i \leq h-1\}$$

and

$$W = \{s_1s_2 \cdots s_{2h-1} \mid s_i \in S_l \text{ for } 1 \leq i \leq 2h-1, s_i s_{i+1} \in H_{2l} \text{ for } 1 \leq i \leq 2h-2\}.$$

(a) $|V|^2/|S_l| \leq |W|$; and

(b) there is a string $r \in S_l$ such that $W$ contains at least $|V|^2/|S_l|^3$ strings of the form $s_1s_2 \cdots s_{2h-1}$, where $s_1 = s_{2h-1} = r$.

**Proof.** (a) Let $S_l = \{r_1, r_2, ..., r_q\}$, i.e., $q = |S_l|$. By $b_i$ ($c_i$) we denote the number of strings in $V$ with prefix (suffix) $r_i$ for $i = 1, 2, ..., q$. Hence $|V| = \sum_{i=1}^{q} b_i$. It is easy to see that $s_1s_2 \cdots s_h \in V$ if and only if $s_h s_{h-1} \cdots s_1 \in V$, and that if $s_1s_2 \cdots s_h \in V$ and $s_h s_{h+1} \cdots s_{2h-1} \in V$, then $s_1s_2 \cdots s_{2h-1} \in W$. Therefore $b_i = c_i$ for each $i$ and
W \geq \sum_{i=1}^{q} c_i b_i = \sum_{i=1}^{q} b_i^2. \text{ By Hölder's (or Cauchy–Bunakowski's) inequality we have } \sum_{i=1}^{q} b_i^2 \geq \left(\sum_{i=1}^{q} b_i\right)^2/q = |V|^2/|S_1|.

(b) By $a_{ij}$ we denote the number of strings in $V$ with prefix $r_i$ and with suffix $r_j$ for $i, j = 1, 2, ..., q$. Let $b_k$ be the maximum number among $b_1, b_2, ..., b_q$. Obviously $V$ contains at least $|V|/q$ strings with prefix $r_k$. Therefore $\sum_{i=1}^{q} a_{kj} = b_k \geq |V|/q$. Since $s_1 s_2 \cdots s_h \in V$ iff $s_h s_{h-1} \cdots s_1 \in V$, $a_{ij} = a_{ji}$ for each $i$ and $j$. Again, by Hölder's (or Cauchy–Bunakowski's) inequality we have that the number of the strings of the form $s_1 s_2 \cdots s_{2h-1}$ such that $s_1 s_2 \cdots s_h \in V$, $s_h s_{h+1} \cdots s_{2h-1} \in V$, $s_1 = s_{2h-1} = r_i$, and $s_h = r_j$ (for $j = 1, 2, ..., q$) is at least $\sum_{j=1}^{q} a_{kj} = (\sum_{j=1}^{q} a_{kj})^2/q \geq |V|^2/|S_1|^3$ and all of them belong to $W$.

**Corollary 3.** For any given $l \in \{2, 4, 8, \ldots, n/4\}$, (a) and (b) below hold for $D_j$, $j = 0, 1, \ldots$, where $D_0 = H_2$, and for $j = 1, 2, \ldots$, $D_j = \{s_1 s_2 \cdots s_{2j-1} \mid s_i \in S_1 \text{ for } 1 \leq i \leq 2j+1, s_is_{i+1} \in H_2 \text{ for } 1 \leq i \leq 2j\}$.

(a) $|D_{j-1}|/|S_1| < |D_j|$; and

(b) for each $j = 1, 2, 3, \ldots$ there is a string $t_j \in S_1$ such that $D_j$ contains at least $|D_{j-1}|^2/|S_1|^3$ strings of the form $s_1 s_2 \cdots s_{2j+1}$, where $s_i = s_{2j+1} = t_j$.

**Lemma 5.** If for some $l \in \{2, 4, 8, \ldots, n/4\}$, $|C_2| < \alpha |S_2|$, then $(1 - 2\alpha)|S_2| < |H_2|$.

**Proof.** Let

$$E_{2l} = \{ss' \in S_2 \mid s, s' \in S_1, ss' \in C_{2l} \text{ and } s's \in C_{2l}\};$$

$$F_{2l} = \{ss' \in S_2 \mid s, s' \in S_1, ss' \in C_{2l} \text{ and } s's \notin C_{2l}\};$$

and

$$G_{2l} = \{ss' \in S_2 \mid s, s' \in S_1, ss' \notin C_{2l} \text{ and } s's \in C_{2l}\}.$$

Clearly $S_{2l} = E_{2l} \cup F_{2l} \cup G_{2l} \cup H_{2l}$, the sets $E_{2l}, F_{2l}, G_{2l}, H_{2l}$ are pairwise disjoint, $E_{2l} \cup F_{2l} = C_{2l}$ and $|F_{2l}| = |G_{2l}|$. Consequently, if $|C_{2l}| < \alpha |S_2|$, then $|E_{2l}| + |F_{2l}| = |C_{2l}| + |F_{2l}| = |C_{2l}| + |F_{2l}| = |C_{2l}| < \alpha |S_2|$ and hence $|H_{2l}| = |S_{2l}| - |E_{2l}| - |F_{2l}| - |G_{2l}| > |S_{2l}| - 2\alpha |S_2| + |E_{2l}| \geq (1 - 2\alpha)|S_2|$. □

Lemma 6 shows that if the set $C_{2l}$ is too small, then the set $D_\alpha$ of Corollary 3 contains a valid input for the algorithm $A$.

**Lemma 6.** If for some $l \in \{2, 4, 8, \ldots, n/4\}$, $|C_{2l}| < \alpha |S_2|$, then there is a string $s_1 s_2 \cdots s_{2l+1} \in D_\alpha$ with $s_1 = s_{2l+1}$, where $p = \log(n/l)$ and $D_\alpha$ is as in Corollary 3, such that each $s_i \in S_1$ and for $1 \leq i < j \leq 2^p$ set($s_i$) $\cap$ set($s_j$) = $\emptyset$.

**Proof.** By $D'_\alpha$, we denote the subset of $D_\alpha$ which contains at least $|D_{\alpha-1}|^2/|S_1|^3$ strings of the form described in (b) of Corollary 3. We are looking for our string among those in $D'_\alpha$. To the contrary, assume that there is no such string in $D'_\alpha$, i.e.,
assume that for each string \( v = s_1 s_2 \cdots s_{2^p+1} \) in \( D_p' \) with \( s_1 = s_{2^p+1} = t_p \), there is a pair of indices \( i, j, 1 \leq i < j \leq 2^p \), such that \( \{s_i\} \cap \{s_j\} \neq \emptyset \). This means that 

\[ |\{s_i\}| \leq |D_p' - 1| \]

But the number of such \( v \)'s (i.e., the cardinality of \( D_p' \)) can be bounded above by \( I \equiv (|D_p' - 1|)^{(2^p + 1)} \sum_{k=1}^{2^p-1} \binom{|L|-1}{k-1} = (n-1)^{\alpha+1} \sum_{k=1}^{2^p-1} \binom{|L|-1}{k-1} \). The factor \( \sum_{k=1}^{2^p-1} \binom{|L|-1}{k-1} \) denotes the number of different sets \( \{s_j\} \cap \{s_i\} \neq \emptyset \) and the factor \( (|D_p' - 1|)^{2^p + 1} \) denotes the number of different functions \( f \) mapping \( \{1, 2, \ldots, |D_p' - 1|\} \) into \( \{1, 2, \ldots, 2^p\} \), where \( f(i) = j \) means that the \( i \)-th member of \( v \) is exactly the \( j \)-th largest element of \( \{s_i\} \). On the other hand \( |D_p'| \geq \frac{|D_p'|}{|S|^3} \) and by repeated applications of part (a) of Corollary 3 followed by a single application of Lemma 5 we have that \( |D_p'| \geq \frac{|D_p'|}{2^p / |S|^3} \geq |H_{2l}|^{2^p / |S|^3} > ((1-2\alpha)^{|L|-1})^{|S|^{2^p + 1}} = (1-2\alpha)^{|L|-1} \cdots (|L|-2l+1)^{|S|(|L|-1) \cdots (|L|-l+1)} = \equiv J \). But our upper and lower bounds on \( |D_p'| \) and \( J \) contradict (4) since \( |L| \geq m_0 \).

**Theorem 2.** Let \( \alpha \) be a real number, \( 0 < \alpha < \frac{1}{2} \), and let \( n \) be a power of 2. Then there is a positive integer \( m_0 \) such that (a) and (b) hold.

(a) If \( L \) is a finite set of labels with \( m_0 \leq |L| \) and \( A \) is a maximum-finding algorithm for a bidirectional ring of size \( n \) labeled by \( L \), then the average number of messages sent by \( A \) is at least \((\alpha/4)n \log n \).

(b) If \( L \) is a finite set of labels with \( m_0 \leq |L| \) and \( A \) is a maximum-finding algorithm for a unidirectional ring of size \( n \) labeled by \( L \), then the average number of messages sent by \( A \) is at least \((\alpha/2)n \log n \).

In the proof below we first observe that the condition of Lemma 6 cannot hold. If it holds, we obtain by the conclusion of the lemma a valid input for the algorithm \( A \) and then show that \( A \) cannot correctly find the maximum given this input. Consequently, all the sets \( C_{2l} \) are relatively large and we can charge \( 1/2 \) messages to segments labeled by elements of \( C_{2l} \). The lower bound is obtained by appropriately summing up these charges.

**Proof of Theorem 2.** (a) Assume \( n \geq 8 \). (The proof is obvious for \( n \leq 4 \).) Let the \( C \)'s and \( S \)'s be defined for any particular \( L \), \( n \), and \( A \) with \( |L| \geq m_0 \), where \( m_0 \) satisfies (4). First we show that \( |C_{2l}| \geq \alpha |S_{2l}| \) for each \( l \in \{2, 4, 8, \ldots, n/4\} \). To the contrary, assume that there is \( l \in \{2, 4, 8, \ldots, n/4\} \) with \( |C_{2l}| < \alpha |S_{2l}| \). Choose \( p \) and \( s_1 s_2 \cdots s_{2^p+1} \) according to Lemma 6. Let a ring of size \( n \) be labeled by \( S = s_1 s_2 \cdots s_{2p} \), with \( s_i = u^1_i u^2_i \cdots u^t_i \) for each \( i = 1, 2, \ldots, 2^p \), and \( u^j_i \in L \) for each \( i \) and \( j \). By Lemma 6, \( S \) is a valid input since all its labels are different. Consider the following computation of \( A \) in such ring. First execute \( c(s_i) \) in the segment labeled by \( s_i \) for each \( i \). Keep the transmission speed of the channels connecting the segments labeled by \( s_1 \) and \( s_2, \ldots, s_{2^p-1} \) and \( s_{2^p} \) and \( s_1 \) very slow during the execution of the \( c(s_i) \)'s.

The concatenation of each pair of consecutive \( s_i \)'s on the ring is in \( H_{2l} \), which implies that for \( i = 1, 2, \ldots, 2^p \) \( |c(s_i s_{i+1})| - |c(s_i)| - |c(s_{i+1})| \leq \frac{1}{2} - 1 \) and the number of messages in \( c(s_i s_{i+1}) \) after the completion of \( c(s_i) \) and \( c(s_{i+1}) \) is smaller
than \(l/2\) (recall that \(s_1 = s_{2^p+1}\)). Hence there is a continuation of the computation that mimics \(c(s_is_{i+1})\) for all \(i\), \(1 \leq i \leq 2^p\), even though the corresponding segments overlap, as no message in the continuation can reach the processors in the middle of the segments labeled by \(s_i\) or \(s_{i+1}\) and no two consecutive continuations can interfere. At the end of the computations \(c(s_is_{i+1})\) for all \(i\) the whole ring is waiting, which means that the computation has terminated. But if the maximum label is in the segment labeled by \(s_i\), this information cannot reach the middle processor of the segment labeled by \(s_{i+1}\) (the processor labeled by \(u_{i+1}'\), say) before the algorithm stops—contradiction. Hence \(|C_{2l}| > \alpha |S_{2l}|\) for \(l = 2, 4, 8, \ldots, n/4\).

We now consider the following computation in the ring labeled by \(r_1r_2r_3\cdots r_n\). Start with the computations in \(c(r_1r_2), c(r_3r_4), c(r_5r_6), c(r_7r_8), \ldots\), then mimic the continuation of \(c(r_1r_2r_3r_4), c(r_5r_6r_7r_8), \ldots\), then mimic the continuation of \(c(r_1r_2\cdots r_8), \ldots\), etc. We call the segments involved special segments. Note that unlike the case above, special segments of the same size do not overlap. We charge a special segment labeled with \(ss'\) the messages sent in \(c(ss')\) after the completion of \(c(s)\) and \(c(s')\). Hence each message sent is only charged once.

We now estimate the total number of messages received during the computations corresponding to all possible assignments of labels to the processors. Let \(l \in \{2, 4, 8, \ldots, n/4\}\) and consider a special segment \(z\) of length \(2l\). There are \(|C_{2l}| > \alpha |S_{2l}| = \alpha |L| (|L| - 1) \cdots (|L| - 2l + 1)\) different ways how to label \(z\) by strings in \(C_{2l}\) and there are \((|L| - 2l)(|L| - 2l - 1) \cdots (|L| - n + 1)\) different ways how to label the rest of the ring. If \(ss' \in C_{2l}\), where \(s, s' \in S_{2l}\), then there are at least \(l/2\) messages received (in \(z\)) after finishing \(c(s)\) and \(c(s')\) and until finishing \(c(ss')\). It means that the special segments of length \(2l\) are charged at least \((n/2l)(l/2) |C_{2l}| \sum_{1 \leq l \leq n/4} \log |L| (|L| - 1) \cdots (|L| - n + 1)\) for each \(l = 2, 4, 8, \ldots, n/4\). For \(l = n/2\) there are \(|L| (|L| - 1) \cdots (|L| - n + 1)\) different ways to label a special segment \(z\) of length \(n\) by strings in \(S_{n/2}\). If \(ss' \in S_{n/2}\), where \(s, s' \in S_{n/2}\), then there are at least \(n/2\) messages received (in \(z\)) after finishing \(c(s)\) and \(c(s')\) until finishing the computation, because each processor must know the maximum label. Summing up (over all \(l\)'s), we have that there are at least \((\alpha/4)n \log |L| (|L| - 1) \cdots (|L| - n + 1)\) message. But the factor \(|L| (|L| - 1) \cdots (|L| - n + 1)\) denotes the number of all possible assignments of labels to the processors. This completes the proof of (a).

(b) The proof for a unidirectional ring is the same as for a bidirectional ring except that \(C_{2l} = \{ss' \in S_{2l}, s, s' \in S_{l}, c(ss') - c(s) - c(s') > l\}\) and to derive the contradiction in the first part of the proof (assuming \(|C_{2l}| < \alpha |S_{2l}|\)) we show that the last processor of the segment labeled by \(s_{i+1}\) (the one labeled by \(u_{i+1}'\)) cannot know the maximum.

Theorem 2 requires a very large label set. \(|L|\) is exponential in \(n\). Recently, Bodlaender [3] proved an \(\Omega(n \log n)\) lower bound for the average case when \(n\) is known with a label set \(L\) satisfying \(|L| > cn\). His elegant proof uses extremal graph theory.
ACKNOWLEDGMENTS

We thank Ran Canetti, Shlomo Moran, Benny Pinkas, and the two referees for helpful suggestions.

REFERENCES