SENSE OF DIRECTION IN COMPLETE DISTRIBUTED NETWORKS

by

Z. Harpaz and S. Zaks

Technical Report #453

April 1987
SENSE OF DIRECTION IN COMPLETE DISTRIBUTED NETWORKS

Z. Harpaz and S. Zaks

Department of Computer Science
Technion, Haifa 32000, Israel

ABSTRACT

In this work we study trade-offs between sense of direction and communication complexity in distributed complete networks. We extend lower bound proofs for complete networks with no sense of direction (presented in [KMZ1, KMZ2]) for other models of these networks. We start with the basic situation, where unused communication lines adjacent to a processor are indistinguishable, and continue through several models, where varying amounts of information are available to each processor, regarding the neighbors across the unused communication lines. We mainly study the problems of minimum-weight spanning tree, leader election, and relations between the models. Our bounds apply to both synchronous and asynchronous communication schemes, and arbitrary message length is permitted.
1. INTRODUCTION

A distributed asynchronous network consists of a set of processors, with distinct initial identities, that are interconnected via a communication network. The processors have to communicate by exchanging messages using the communication lines adjacent to them, in order to solve a certain problem. Messages arrive without error after a finite (but unbounded) delay. All the communication lines adjacent to a processor are indistinguishable; in other words, it has no sense of direction (IS). The more knowledge a processor is given about these communication lines, the less communication activities might be needed, and this study was motivated by this trade-off between sense of direction and communication complexity.

Two most frequently studied problems in distributed networks are the problem of constructing a minimum-weight spanning tree (MST) and the problem of distinguishing a unique processor (a leader, a problem very closely related to the spanning tree (ST) problem). The two commonly studied symmetric network topologies that recently drew most attention are the circular network and the complete network.

The circular network is the simplest symmetric network to work with, both practically and theoretically. It is thoroughly studied, and lower and upper bounds for the election problem are known for various models (see, for example, [B, DKR, FL, HS, LT, MSZ; P, PKR]). Sense of direction in a bidirectional ring is referred to as the existence of a fixed known direction on the ring (as defined, for example, in [B]). The impact of sense of direction in a ring is still not fully understood.

Another studied symmetric topology is the complete network. In [AG, KMZ1] it is shown that \(\Theta(n \log n)\) messages are sufficient (and in the worst case necessary) to elect a leader in a complete network with no sense of direction. Now, suppose we arrange the processors on a ring (numbered 1, 2, \ldots, \(n\); these numbers are not known to the processors), such that processor \(i\) knows which of its adjacent edges connects it with processor \(i+1, i+2, \ldots, i-1\) (modulo \(n\)). This is a case where a processor has a global sense of direction. In [LMW] it is shown that in this case a ST can be constructed using at most \(O(n)\) messages; actually, an \(O(n)\) algorithm can be achieved with only partial sense of direction (for example, in the case where the processors are arranged in a ring and only a set of \(O(\log n)\) of the edges adjacent to each processor are oriented), as studied in [ASZ], where it is also shown that this result is optimal. In [KMZ2] a tight lower bound of \(\Omega(n^2)\) messages is presented, for constructing an MST in complete graphs with no sense of direction.
In this work we study trade-offs between sense of direction and communication complexity for several problems in distributed complete networks. We mainly consider two models that use amounts of information that lie between the two extremes of no sense of direction and global sense of direction, and mainly study the communication complexity of the election and MST problems, by extending the lower bounds proofs presented in [KMZ1, KMZ2]. We also study few other problems, that are related to the impact of sense of direction on communication complexity in complete networks.

In section 2 we define these models. Constructions of the models and relations between them are studied in section 3. We then scan through the problems, and discuss their complexities in the various models. Results concerning election, MST and cycle partition problems are the subject of sections 4, 5 and 6, respectively. Miscellaneous problems are the subject of section 7.

2. PRELIMINARIES

The model under investigation is a network of n processors with distinct identifiers. Each processor has some communication lines, connecting it to some others (to all others in the case of a complete network). The processors have to solve a certain problem by exchanging messages along the communication lines. The processors all perform the same algorithm, that includes operations of sending a message to a neighbor, receiving a message from a neighbor and processing information in their (local) memory. We assume that the messages arrive, with no error, after a finite but otherwise arbitrary delay, and are kept in order in a queue until processed. We also assume that any non-empty set of processors may start the algorithm; a processor that is not a starter remains asleep until a message reaches it. An execution of an algorithm consists of a sequence of events of sending and receiving messages according to the algorithm. An algorithm may have more than one execution due to the unpredictable starting processors and message delays. We identify the network and the complete graph representing it. Therefore, the terms processor and vertex will be used interchangeably, and so will communication line and edge. An edge is used if it carried a message, and it is unused otherwise. The message complexity of an algorithm is the maximum number of messages the algorithm may send during any possible execution. No assumption is made about the length of a message, unless otherwise specified. The edge complexity of an algorithm is the maximum number of used edges upon the completion of any execution of the algorithm. The processor's knowledge of where the communication lines are leading to determines the amount of sense of direction it has (we define several options later). All our lower bounds deal with edge complexity, and therefore apply also to
message complexity.

We now define the models studied in this work. The numbers 1, 2, ..., n are associated with the processors; these numbers are not known to the processors, and we use them in order to ease the discussion. With the communication line \((i,j)\) connecting processor \(i\) and processor \(j\) we associate a label \(l(i,j)\), known only to processor \(i\). Denote \(N_i = \{1,2,\ldots,n\} - \{i\}\) and \(N' = \{1,2,\ldots,n-1\}\). The following models are defined:

[G] The **general model**: for every \(i\)

\[
\{ l(i,j) \mid j \in N_i \} = N'.
\]

This is the basic and most common model and it represents the case where no sense of direction is provided to the processors. (See example for \(n=6\) in Figure 1.)

[C] The **complement model**: for every \(i\)

\[
\{ l(i,j) \mid j \in N_i \} = N'
\]

and for every \(i\) and \(j\)

\[
l(i,j) + l(j,i) = n.
\]

This model was introduced in [SUZ]. (See example for \(n=6\) in Figure 2.)

[M] The **matching model**: for every \(i\)

\[
\{ l(i,j) \mid j \in N_i \} = N',
\]

and for every \(i\) and \(j\)

\[
l(i,j) = l(j,i).
\]

Note that in such a labeling all the edges labeled with \(i, i \in N'\), form a perfect matching. Therefore such a labeling might exist only for an even \(n\); an extension for the odd case is straightforward. (See example for \(n=6\) in Figure 3.)

[CR] The **chordal ring model**: for every \(i\) and \(j\)

\[
l(i,j) = (j-i) \mod n.
\]

Note that in this model for every \(i\)

\[
\{ l(i,j) \mid j \in N_i \} = N',
\]

and for every \(i\) and \(j\)

\[
l(i,j) + l(j,i) = n.
\]

This model was introduced in [LMW]. (See example for \(n=6\) in Figure 4.)
Figure 1. The general model.

Figure 2. The complement model.
Figure 3. The matching model.

Figure 4. The chordal ring model.
The corresponding labelings in these models are termed \( G \) labeling, \( C \)-labeling, \( M \) labeling, and \( CR \) labeling, respectively. Note that every \( CR \) labeling is also a \( C \) labeling, and that every \( CR \), \( C \) or \( M \) labeling is a \( G \) labeling. These relations are depicted in Figure 5.

---

**Figure 5. Relations between the models.**

---

These models represent varying degree of sense of direction a processor has about the network. The general model does not provide the processor with any information. In the complement model each edge has a unique label that depends on the label at the other end of the edge. Therefore, unused edges are distinguishable, but there is no global arrangement of the processors, and at each processor the exact orientation of the edges is unknown. The matching model is similar to the complement model, with the additional fact that all the edges labeled \( k \) form a perfect matching. In the chordal ring model we say that a *global sense of direction* is implied by the labels; the processors are arranged in a ring (the edges labeled 1 and \( n-1 \)) and at each processor the edge labeled \( k \) is connected to the \( k \)-th processor on a fixed known direction on the ring.

A basic operation that is used throughout the discussion is exchanging labels of unused adjacent edges. For this we use the following assumption: suppose a given execution of algorithm \( A \) terminated with two unused edges \( (v,u) \) and \( (v,w) \) (Figure 6(a)). If we disconnect these edges (see Figure 6(b)), then the same execution can be repeated, with no processor being able to detect the difference. Now, if we reconnect these edges, but exchange their endpoints (as depicted in Figure 6(c)), then the same execution can take place and no processor can detect the difference. We refer to this property as the *basic exchange property*. Exchanging endpoints of two adjacent edges, or exchanging their labels, is actually the same opera-
tions, and we will exchange labels and justify it with the above operation.

Figure 6. The basic exchange operation

3. RELATIONS BETWEEN THE LABELINGS

In this section we study basic relations between the labelings, by presenting \( \Omega(n^2) \) lower bounds for certain problems. All these bounds are tight since the reader can easily construct algorithms that solve these problems using \( O(n^2) \) messages.

3.1. Verification

The verification problem is the following: given labels to the edges at each processor, determine whether these labels form a given labeling.

Theorem 1: The edge complexity of determining whether a given labeling is a \( C, M \) or \( CR \) labeling, is at least \( \Omega(n^2) \).

Proof: For contradiction, assume that there exists an algorithm \( A \) which determines whether a given labeling is a \( C, M \) or \( CR \) labeling, using \( o(n^2) \) edges. Then there are three distinct vertices, \( v, s \) and \( t \), such that the edges \( (v,s) \) and \( (v,t) \) are unused after an execution of \( A \) on a network which is labeled with a \( C, M \) or \( CR \) labeling. If we exchange the labels \( I(v,s) \) and \( I(v,t) \), and apply \( A \) again, it might produce the same positive answer (by the basic exchange property). But now the labeling is no longer a \( C, M \) or \( CR \) labeling, a contradiction.
same positive answer (by the basic exchange property). But now the labeling is no longer a C, M or CR labeling, a contradiction.

Q.E.D.

3.2. Transformation

A transformation between two labelings is constructing one of the labelings, given the other, with a minimum number of label changes. We first extend a technique from [KMWZ2] to derive the following

Theorem 2: The edge complexity of transforming an M labeling to a C labeling, or transforming a C labeling to an M labeling, is at least $\Omega(n^2)$.

Proof: First we prove for transforming an M labeling to a C labeling. We show two different matching labelings such that the sets of labels that have to be changed in order to obtain corresponding complement labelings must be distinct. Thus every transformation algorithm must distinguish between the two labelings, and in order to do so we show that it must send messages on at least $\Omega(n^2)$ edges.

Let $K_n = (V, E)$ be the complete undirected graph on $|V| = n$ vertices and let $n = 4 \text{ mod } 12$. A vertex division $\langle V_1, V_2, \ldots, V_{\frac{n}{4}} \rangle$ of $V$ is a partition of $V$ into pairwise disjoint sets of four vertices each; namely, $V = V_1 \cup V_2 \cup \cdots \cup V_{\frac{n}{4}}$, where $|V_i| = 4$ for all $i$ and $V_i \cap V_j = \emptyset$ for all $i \neq j$. The graph $G'$ induced by a vertex division $\langle V_1, V_2, \ldots, V_{\frac{n}{4}} \rangle$ is $G' = (V, E')$, where $E' = \{\{i, j\} | \exists k: i, j \in V_k\}$. An edge division $\langle E_1, E_2, \ldots, E_{\frac{n}{3}} \rangle$ of $E$ is a partition of $E$ into pairwise disjoint sets $E_i$ such that each graph $G_i = (V, E_i)$ is induced by a vertex division. For the proof we need the following:

Theorem ([HCW]): The complete graph $K_n$ has an edge division if and only if $n = 4 \text{ mod } 12$.

Now let $n = 8 \text{ mod } 24$. Partition $V$ into two disjoint sets $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$, where $m = \frac{n}{2}$. Note that $|X| = |Y| = 4 \text{ mod } 12$. Let $G_1 = (X, E_1)$, and $G_2 = (Y, E_2)$ be the subgraphs induced by $X$ and $Y$, respectively. Let $\langle A_1, A_2, \ldots, A_{\frac{n}{3}} \rangle$ be an edge division of $E_1$; $\langle B_1, B_2, \ldots, B_{\frac{m}{3}} \rangle$ is obtained from $\langle A_1, A_2, \ldots, A_{\frac{n}{3}} \rangle$ by replacing every $x_i$ by $y_i$. We define two
matching labelings:

First matching labeling ($l_1$):

Each $A_i$ and $B_i$ consists of disjoint 4-cliques. Let $x_a, x_b, x_c$ and $x_d$ ($y_a, y_b, y_c$ and $y_d$), $a < b < c < d$, be vertices of a 4-clique that belongs to $A_i$ ($B_i$). Since in a matching labeling $l(a, b) = l(b, a)$ for all $a$ and $b$, it is sufficient to define it for $a < b$. We define $l_1$ as follows:

\[
\begin{align*}
  l_1(x_a, x_b) &= l_1(y_a, y_b) = l_1(x_a, x_d) = l_1(y_a, y_d) = 3i - 2, \\
  l_1(x_b, x_d) &= l_1(y_b, y_d) = l_1(x_b, x_c) = l_1(y_b, y_c) = 3i - 1, \\
  l_1(x_c, x_d) &= l_1(y_c, y_d) = l_1(x_c, x_a) = l_1(y_c, y_a) = 3i,
\end{align*}
\]

for $1 \leq i < j \leq m$,

\[
l_1(x_i, y_j) = \begin{cases} \\
  l_1(y_i, x_j) = n - l_1(x_i, x_j), & \text{for every } 1 \leq i, j \leq m. \\
  l_1(x_i, y_j) = n._{m} & \text{else.}
\end{cases}
\]

The labeling $l_1$ of two corresponding 4-cliques for $n = 32$ and $i = 3$, is depicted in Figure 7.

Second matching labeling ($l_2$):

Let $x_a, x_b, x_c, x_d, y_a, y_b, y_c$, and $y_d$ be as above. The second labeling $l_2$ is derived from $l_1$ by changing the labels of the edges induced by exactly one set $\{x_a, x_b, x_c, y_a, y_b, y_c, y_d\}$ in the following way (denote $x_a, x_b, x_c$, and $x_d$ ($y_a, y_b, y_c$, and $y_d$) belong to $A_i$ ($B_i$)). Define $l_2$ as follows:

\[
\begin{align*}
  l_2(x_a, x_d) &= l_2(y_a, y_d) = l_2(x_a, x_c) = l_2(y_a, y_c) = 3i_0 - 2, \\
  l_2(x_b, x_d) &= l_2(y_b, y_d) = l_2(x_b, x_a) = l_2(y_b, y_a) = 3i_0 - 1, \\
  l_2(x_c, x_d) &= l_2(y_c, y_d) = l_2(x_c, x_b) = l_2(y_c, y_b) = 3i_0, \\
  l_2(x_a, x_d) &= l_2(y_a, y_d) = l_2(x_a, x_c) = l_2(y_a, y_c) = n - (3i_0 - 2), \\
  l_2(x_b, x_d) &= l_2(y_b, y_d) = l_2(x_b, x_a) = l_2(y_b, y_a) = n - (3i_0 - 1), \\
  l_2(x_c, x_d) &= l_2(y_c, y_d) = l_2(x_c, x_b) = l_2(y_c, y_b) = n - 3i_0.
\end{align*}
\]

and

\[
l_2(i, j) = l_1(i, j)
\]

for all other cases. The 8-clique changed in $l_2$ will be denoted by $G(l_2)$. Figure 8 shows the labels that were changed in $l_2$ for the 8-clique of Figure 7 (namely $i_0 = 3$).

Let $G$ be labeled with $l_1$. At each set $\{x_a, x_b, x_c, x_d, y_a, y_b, y_c, y_d\}$ as above the edges labeled $3i - 2$ and
Figure 7. The first labelling $I_1$.

Figure 8. The second labelling $I_2$. 
their complements, (edges labeled \( n - (3i-2) \)) constitute two edge-disjoint cycles, as depicted in Figure 9. The edges labeled \( 3i-1, 3i \) and their complements, respectively, constitute similar cycles. In order to transform \( \tilde{I}_1 \) into a complement labeling, the following actions must be taken on each of the above cycles: two vertices at distance two apart have to exchange the labels (as known to them) of the edges adjacent to them of the cycle. For example, the labelings on the left cycle of Figure 9 will be changed to one of the labelings shown in Figure 10. If \( G \) is labeled with \( I_j \), the edges labeled \( 3i_{(j-1)}, 3i_{(j-2)}, 3i_{j-2} \) and their complements, respectively, still constitute cycles as before, but the vertices in the corresponding cycles in \( G(I_j) \)

Figure 9. The 4-cycles in \( I_1 \).

Figure 10. The resulting complement labels of \( I_1 \).
are ordered differently. Every choice of two vertices in \( G(l_2) \), that will exchange their labels when \( G \) is labeled with \( l_2 \), yields a faulty choice when \( G \) is labeled with \( l_2 \) (a choice is faulty if it causes two adjacent vertices, on a certain cycle, to exchange their labels). In this case, the exchange of the labels will not yield a complement labeling since the edge connecting these two vertices will have the same label at its two endpoints; for example, if \( x_a, y_a, x_e, y_e \) where chosen for the labels \( 3i-2 \) and \( n-(3i-2) \), then with \( l_2, \tilde{x}_e \) and \( \tilde{y}_a \) are adjacent on a \( 3i-2, n-(3i-2) \) cycle: \( l_2(\tilde{x}_e, \tilde{y}_a) = l_2(\tilde{x}_e, \tilde{y}_a) = n-(3i-2) \).

By checking all the possible transformations on \( G \) we can show that every choice of vertices for a transformation from the matching labeling \( l_1 \) to a complement labeling is a faulty choice if the transformation is applied on \( G \) with \( l_2 \).

Let \( A \) be an algorithm for transforming a matching labeling to a complement labeling. \( G' \) \( (G'') \) be a complete graph labeled with \( l_1 \) \( (l_2) \), respectively, and let \( e_1 \) be an execution of algorithm \( A \) that does not send any messages on the edges of \( G(l_2) \). Then by the basic exchange property applying \( A \) on \( G' \) and on \( G'' \) might yield the same results. But this contradicts the previous discussion (saying that no two such resulting labelings can be equal).

We conclude that in every transformation, messages have to be sent on at least one edge of every possible subgraph \( G(l_2) \) (excluding the edges labeled \( \frac{n}{2} \)). Since these \( G(l_2) \)'s are edge-disjoint (excluding the edges labeled \( \frac{n}{2} \)), and there are \( m-1 \times m = \frac{n(n-2)}{48} \) of them, we obtain a lower bound of \( \Omega(n^2) \) edges used by any transformation algorithm.

Given a general \( n \), we label a clique of size \( n' \) with \( l_1 \), where \( n' \) is the largest integer for which the following inequalities hold: \( n' \equiv 8 \mod 24 \) and \( n' \leq \frac{n}{2} \). Then we extend the resulting labeling to an \( M \) labeling over the whole graph. This can be done according to a theorem from [Hi] stating that an edge coloring of \( K_m \) can be extended to an edge coloring of \( K_n \) if \( n \geq 2m \).

The proof of the reverse transformation is very similar to the previous case (starting with two complement labelings corresponding to the matching labelings \( l_1 \) and \( l_2 \)). This completes the proof.

Q.E.D.
In the next section we show a simpler proof to a stronger theorem. However, we presented the last theorem and proof because of the technique it uses, we believe that similar constructions - extending the basic idea of [KMZZ], will lead to lower bounds for additional problems.

3.3. Construction

Constructing a labeling is assigning a label to each edge at each processor, such that a labeling of a given model is formed. In [SUZ] it is shown that the edge complexity of constructing a $C$ labeling (hence also $CR$, labeling), given a $G$ labeling is $\Omega(n^2)$.

The $M$-labeling does not reduce the complexity of constructing a $C$ labeling:

Theorem 3: The edge complexity of constructing a $C$ labeling, given an $M$-labeling, is at least $\Omega(n^2)$.

Proof: Let $n = 4 \pmod{12}$ (an extension to general $n$ is as in the proof for Theorem 2). We use a Theorem (that we mentioned before) from [HGW] to partition the edges of the graph into $n-1\over 3$ pairwise disjoint sets of equal size, each of which induces $n/4$ cliques of size four. We define the following matching labeling:

Each set of edges is associated with three unique colors, $\alpha$, $\beta$ and $\gamma$, and the edges of each clique are colored as depicted in Figure 11. For contradiction, assume that there exists an algorithm that solves the problem, using only $\mathcal{O}(n^2)$ edges. Then there exists a 4-clique, $a$, $b$, $c$ and $d$, that all its edges are unused upon the completion of an execution of the algorithm. By the basic exchange property, the same execution can take place also when the edges of that clique are colored as depicted in Figure 11.2 and 11.3. Denote these cases (1), (2) and (3), respectively. Assume that $a$ replaced the label $\alpha$ with $\gamma$ (where $\alpha = n/2$);

![Diagram](image_url)

(1) (2) (3)

Figure 11. The three cases.
otherwise, take $b$). Then because of case (1) $b$ replaced it with $n - l$. So did $c$ because of case (2). But in case (3) $b$ and $c$ are connected with an edge labeled $\alpha$, and therefore the algorithm does not produce a $C$ labeling in this case. This contradicts our assumption, and completes the proof. Actually this proof holds for construction of any labeling that has different labels on the edges, given an $M$ labeling.

Q.E.D.

In a way similar to the proof of Theorem 1 one can show the following:

Theorem 4: The edge complexity of constructing an $M$ labeling, given a $G$ labeling, is at least $\Omega(n^2)$.

We use the construction from the proof for Theorem 2 to show:

Theorem 5: The edge complexity of constructing an $M$ labeling, given a $C$ labeling, is at least $\Omega(n^2)$.

Proof: Let $n = 8(\text{mod} 24)$ (an extension for general $n$ is as in the proof for Theorem 2). Consider the following $C$ labeling: partition the edges to 8 cliques (as in the proof for Theorem 2) and label each clique as $G(l_8)$. Transform this labeling to a $C$ labeling (that corresponds the labeling of the 4-clique that is depicted in Figure 12.1). For contradiction, assume that there exists an algorithm that solves the problem, using only $o(n^2)$ edges. Then there exists a 4-clique, $x_b$, $x_c$, $y_c$, and $y_d$, that all its edges are unused upon the completion of an execution of the algorithm. By the basic exchange property, the same execution can take place also after the labels of the edges of that 4-clique are changed as depicted in Figure 12. Denote these cases (1) and (2), respectively. Assume $x_b$ replaces the label $l$ with $\alpha$. Then because of case (1) $x_c$ must replace $n - l$ with $\alpha$, and $y_d$ must replace $l$ with a label $\beta \neq \alpha$. But in case (2) the edge $(x_c, y_d)$ is labeled with different labels: $\alpha$ by $x_c$, and $\beta$ by $y_d$. This contradicts our assumption, and implies the theorem.

Q.E.D.

Consider now constructing an $M$ labeling, given a $CR$ labeling. Although the matching model has "less" amount of sense of direction than the chordal ring model, construction of an $M$ labeling demands some communication activity in order to reach a certain amount of coordination between the processors: consider any edge in the graph (that has different labels of its endpoints). The processors on both ends have to assign to it the same label. But a $CR$ labeling is symmetric, and this edge is a different edge from the
local point of view of each of the processors. Therefore each one of them acts differently, and they have to coordinate their action. Hence, every processor that wakes up must send a message. On the other hand, a leader, using \( n \) messages can coordinate all the other processors. Thus we have:

**Corollary 1:** The message complexity of constructing an \( M \) labeling, given a \( CR \) labeling, is \( \Theta(n) \).

### 4. ELECTION

Electing a leader is distinguishing exactly one processor from the others. The two problems of election and construction of a spanning tree are closely related to each other, and are of the same complexity. Both problems require a global algorithm. (A global algorithm (see, [KMZ1]) is an algorithm, in all of which executions the used edges induce a spanning subgraph.) We name this section by the election problem because of its importance; however, all our bounds apply to any global algorithm. In [KMZ1] it was shown that the complexity of global algorithms given a \( G \) labeling is \( \Theta(n \log n) \) messages. On the other hand (see [LMW]), this problem has complexity of \( \Theta(n) \) messages given a \( CR \) labeling.

#### 4.1. Matching labeling

The main difficulty of a global algorithm in a complete network is how to avoid unnecessary messages to those processors that are already “found” which is the essence of the \( \Omega(n \log n) \) lower bound in [KMZ1]. We extend the technique from [KMZ1] to show that the matching model does not help in
reducing the complexity of global algorithms. Namely:

**Theorem 6:** The edge complexity of a global algorithm, given an $M$ labeling, is at least $\Omega(n \log n)$.

**Proof:** The proof is based on the proof for the general model in [KMZ1], However the main claim there, i.e. unused edges are indistinguishable, does not hold in this case, and is modified.

First we note that if $A$ is a free distributed algorithm that solves a certain problem, whose message complexity is a function of $n$, then there exists a comparison algorithm that solves the same problem, with the same message complexity ([FL]). Therefore, we restrict the proof to comparison algorithms. Since through this discussion we relate to the labels as edge colors, we also denote them as such. A color becomes used when an edge colored with it does. We use an adversary that acts as follows:

1. Initially, all the edges are not connected (i.e., every processor has all its edges, colored with $n-1$ distinct colors, but these edges are not yet connected at the other side). During the execution the adversary connects two unconnected endpoints labeled with the same color to form an edge. (This reflects the fact that in the matching model a processor doesn’t know anything about the processor that is on the other side of an unused edge, apart from the color of this edge at that processor.) $m$ processors are awakened where $m = 2^k \leq \frac{n}{2} < 2^{k+1}$.

2. The adversary works in phases. At the beginning of the $i$-th phase, $i = 1, \ldots, k$, connected components (in the graph induced by the awakened vertices and the edges used so far) of size $2^{i-1}$ are matched to form half number of double sized components. During the phase, when a message is sent on an "unconnected" edge, the adversary connects all the edges labeled with the same label in the component, i.e. $2^{i-1}$ edges (we prove later that such a message always exists). The phase ends when in each component $2^{i-1}$ such messages are sent, and all the edges of the component are connected. At the last phase, a component of size $m$ is formed.

3. The adversary connects all the remaining unconnected edges, in such a way that yields an $M$ labeling (and the algorithm proceeds until its completion).

Now we have to show that the above scenario is feasible.

1. At each connected component the algorithm behaves the same. This is due to the fact that the algorithm is a comparison algorithm and that the adversary can connect the edges in the same way at
2. At every phase \( i \) and at every component of size \( 2^i \) the algorithm uses \( 2^{i-1} \) new edges. At each component, there will eventually be a vertex that will send a message on an unused edge, until the used edges span the whole graph ([KMZ1]). Whenever such edge is used the adversary connects \( 2^{i-1} \) edges of the same color within the component. This operation can be repeated \( 2^{i-1} \) times due to König’s theorem [K], stating that \( K_m \) can be colored with \( t \) colors, and the fact that the components are similar. Note that when a message is sent on an unconnected edge labeled \( i \), then all the edges labeled \( i \) are unconnected. Therefore, the algorithm is forced to use that number of edges, at each component, before the component grows.

3. The connected edges as described above yield an \( M \) labeling. During the phases the adversary creates an edge coloring of the subgraph induced by the \( m \) awakened vertices, using \( m-1 \) colors. By [Hi] this edge coloring can be extended to a coloring of the whole graph, i.e., an \( M \) labeling.

At phase \( i \) there are \( \frac{m}{2^i} \) connected components. Thus, the number of edges used by the algorithm is at least

\[
\sum_{i=1}^{\log m-1} \frac{m}{2^i} \times 2^{i-1} \geq \frac{m}{2} \times (\log m - 1) = \Omega(m \log m) = \Omega(n \log n),
\]

and the theorem follows.

Q.E.D.

4.2. Complement labeling

We now show:

**Theorem 7:** The message complexity of a global algorithm, given a \( C \) labeling, is at least \( \Omega(n \log n) \).

**Proof:** The proof is for even \( n \). It is very similar to the one in the previous case, and we only outline it.

We use a similar adversary argument; that differs from the previous as follows:

1. At the beginning of the algorithm \( m \) processors (as before) are awakened, each sending a message using the same labeled edge. The adversary connects these edges so that cycles of length four are constructed. In this way, after the first phase there are components of size four and the same two labels are used at all of the awakened processors.
For each new label (first use of the label in the component) that the algorithm uses, the adversary connects all the edges labeled $l$ and $n-l$ at all the component's vertices. (i.e., two edges at each vertex, instead of one in the matching model case. Note that the connected edges belong to a bipartite subgraph and therefore form disjoint cycles of even lengths.) In this way, if a label is used at some vertex, then all the edges labeled with it are connected throughout the component. Thus, when a message is sent on an unconnected edge, this edge is labeled with a new label, and the adversary can proceed the same way it does in the matching model case. The adversary then forces the algorithm to use $\frac{k}{4}$ new edges in each component of size $k$, at each phase.

At the end of the last phase (i.e., after $\frac{m}{4}$ edges were used at the component of size $m$) the adversary:

(3.1) Transfers the partial complement labeling to a (partial) matching labeling. (This is possible due to the even length cycles mentioned before.)

(3.2) Extends the coloring of the edges to the whole graph ([HIl]).

(3.3) Transfers the labeling back to a $C$ labeling, so that the labels of edges that were connected before the transformations, are the same as before. This is feasible because of the following: all the edges labeled with the same labels create disjoint cycles, and a conflict between the original $C$ labeling and the new one can rise only in such cycles that contain labels from both labelings. All the edges that were connected before the transformations cover the same labels at each one of the awakened vertices. Therefore there is no such cycle (that contains labels from both labelings).

The counting of edges that the algorithm uses is similar to the one in the matching model case, except that (1) the second phase does not exist, and (2) only half the number of edges is used during a corresponding phase. Combining these changes with the proof for the $M$ labeling case yields the proof.

Q.E.D.

5. MINIMUM-WEIGHT SPANNING TREE CONSTRUCTION

Constructing an $MST$ is the following problem: given a weight to each edge, known only to its endpoints, each processor has to select some of its adjacent edges such that each edge is selected by both its
endpoints and all of the selected edges form an MST. In [KMZ2] a lower bound of $\Omega(n^2)$ edges was shown for the MST problem given a $G$ labeling. This bound is tight due to the algorithm that was presented in [GHS], which requires $\tilde{O}(n \log n + 1E)$ messages. We extend the proof in [KMZ2] to show that the information implied by the matching and complement models does not help to reduce the complexity of the MST problem.

5.1. Matching labeling

Theorem 8: The edge complexity of the MST problem, given an $M$ labeling, is at least $\Omega(n^2)$.

Proof: The proof is based on the fact that although unused edges are distinguishable (due to their distinct labels), a processor still cannot tell whether their other endpoints are exchanged. We show two different graphs, each having an MST with different weight. Thus, in order to construct an MST every algorithm must distinguish between these graphs and in order to do so we show that it must send messages on at least $\Omega(n^2)$ edges.

Let $G$ be a complete graph $K_n$ labeled with $1$ (as defined in the proof for Theorem 2). The first graph $\tilde{G}$ is constructed from $G$ by assigning weights to its edges as follows: the edges in $E_1$ and $E_2$ (as defined in the proof for Theorem 2) get a 0 weight and all the other edges of $G$ (i.e., $\{(x,y) | x \in X, y \in Y\}$) get a 1 weight. The second graph $\tilde{G}$ is constructed from $\tilde{G}$ by exchanging the labels and weights of four edges as follows: let $x_a, x_b, y_a, y_b$ be a cycle in $\tilde{G}$. The new labels and weights are (as depicted in Figure 13; the numbers inside (outside) the squares denote the weights (labels)):

\[
\begin{align*}
\tilde{\tilde{w}}(\tilde{x}_a, \tilde{x}_b) &= \tilde{w}(\tilde{x}_a, \tilde{x}_b), \\
\tilde{\tilde{w}}(\tilde{x}_a, \tilde{y}_b) &= \tilde{w}(\tilde{x}_a, \tilde{y}_b), \\
\tilde{\tilde{w}}(\tilde{y}_a, \tilde{x}_b) &= \tilde{w}(\tilde{y}_a, \tilde{x}_b), \\
\tilde{\tilde{w}}(\tilde{y}_a, \tilde{y}_b) &= \tilde{w}(\tilde{y}_a, \tilde{y}_b), \\
\tilde{\tilde{w}}(\tilde{x}_a, \tilde{x}_b) &= \tilde{w}(\tilde{x}_a, \tilde{x}_b) = 1, \\
\tilde{\tilde{w}}(\tilde{x}_a, \tilde{y}_b) &= \tilde{w}(\tilde{x}_a, \tilde{y}_b) = 0.
\end{align*}
\]

Let $A$ be an algorithm for constructing an MST, and $e$ an execution of $A$ on $\tilde{G}$ that does not send any messages on the edges of the above cycle. The same execution is also possible on $\tilde{G}$, according to the basic exchange property. But this is a contradiction since the weight of an MST of $\tilde{G}$ is 1, while the weight of an MST of $\tilde{G}$ is 0.
Thus, in order to construct an MST every algorithm must send messages on at least one edge of every one of the above cycles. But there are $\Omega(n^2)$ such disjoint 4-cycles (see [KMZ2]), which implies the lower bound of $\Omega(n^2)$ edges. An extension for general $n$ is as in the proof for Theorem 2.

Q.E.D.

5.2. Complement labeling

In a similar way it can be shown that the same holds for the complement model:

Theorem 9: The edge complexity of the MST problem, given a $C$ labeling is at least $\Omega(n^2)$.

Proof: Consider $\tilde{G}$ and $\tilde{G}$ (from the proof of Theorem 8) after their labelings are transformed into $C$ labelings (any transformation will do). The two graphs differ only in the weights and labels of one 4-cycle, as depicted in Figure 14. The rest of the proof is identical to the previous one:

Q.E.D.

5.3. Chordal Ring labeling

Consider now the chordal ring model. The global sense of direction that is implied by the labeling saves the need to use many edges, in order to determine where they lead to, but it does not necessarily help
in cases that require large amount of information transfer. If we allow long messages (of \(O(n \log n)\) bits), then the message complexity of the MST problem, given a CR labeling is \(\Theta(n)\). The lower bound is obvious because the algorithm is global. For the upper bound consider the following simple algorithm:

1. Elect a leader.
2. All the processors send the leader the weights of all their adjacent edges.
3. The leader calculates the MST and sends each processor the labels of the edges of the tree that are adjacent to it.

We conjecture that if the messages are restricted to \(O(\log n)\) bits then the message complexity is \(\Omega(n^2)\). We feel that, in order to prove it, bit complexity techniques are needed, rather than the ones used so far in our discussion.

6. PARTITION INTO CYCLES

Consider now the problem of partitioning into cycles (PC): partition the processors into cycles (of any size greater than 2) such that every processor will know the two edges adjacent to it that belong to the cycle.

We first note that the edge complexity of the PC problem, given a G labeling, is at least \(\Omega(n^2)\). To see this we use the adversary as in [KMZ21]: assign numbers from 1 to \(n\) to the processors, and whenever a processor sends a message on a new edge, deliver this message to the smallest numbered processor possible.
At the end of the algorithm, the used edges contain a matching of at least \( \frac{1}{3} n \) (this bound is achieved in the case where all the cycles are of length 3; any larger cycle increases the size of the matching). Arrange the pairs of the contained matching \((v_i, u_i)\), such that \( v_i < u_i \) and \( v_i < v_{i+1} \). So we have \( v_i \geq i \), and therefore the number of edges the algorithm uses is at least

\[
\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} i = \frac{1}{18} (n^2 + 3n) = \Omega(n^2)
\]

We now show that the amount of sense of direction implied by the C or the M model reduces the complexity of the PC problem. Note that in these models the labels \( i \) and \( n < i \) (for any \( i \)) partition the vertices into disjoint cycles. Therefore, no communication is needed apart from awakening all the vertices. Since a global algorithm is needed, it follows that:

**Corollary 2**: The message complexity of the PC problem, given a C or an M labeling, is \( \Theta(n \log n) \).

Given a CR labeling, the same arguments as above (and considering [LMW]) yields the following:

**Corollary 3**: The message complexity of the PC problem, given a CR labeling, is \( \Theta(n) \).

7. **MORE PROBLEMS**

In this section we present three additional results.

7.1. **The k-partition problem**

The problem: given an integer \( k \) \((k \mid n)\), partition the set of processors into \( \frac{n}{k} \) sets of size \( k \) each, such that at each set there exists a processor that knows the identities of all the other members of the set, and the edge that leads to each of them.

**Theorem 10**: The edge complexity of the \( k \)-partition problem, given a \( G \) labeling, is at least \( \Omega(\max(n \log n, \frac{n^2}{k})) \).

**Proof**: The proof is similar to the proof of the lower bound for the degree-restricted spanning tree in [KMZ], using an adversary argument. The processors are numbered from 1 to \( n \). Whenever processor \( i \) sends a message on a new edge, the adversary will deliver the message to the processor with the smallest
possible number. Let \( t = \frac{n}{k} > 1 \), and denote by \( a_1, a_2, \ldots, a_t \) the processors that have the identities of all the other processors in their set. Upon the termination of the algorithm there are at least \((k-1)t\) used edges. (If two edges that suppose to lead to two set members are unused, then by the basic exchange property their endpoints might be exchanged.) At least half of these edges carried their first message from a processor \( a_i \) or to a processor \( a_j \).

Case 1. At least half of the used edges carried their first message from a processor \( a_i \) to another processor in the set.

Let \( m_i \) be the number of the largest numbered processor in the set of all processors \( p \) such that the edge connecting it to \( a_i \) carried its first message from \( a_i \) to \( p \). Therefore the number of edges used by \( a_i \) is at least \( m_i \). Let \( m_i \geq m_{i+1} \geq \cdots \geq m_{t} \). Then we can write

\[
m_i \geq \left( \frac{1}{2} - j + 1 \right)(k-1) \quad \text{for } 1 \leq j \leq \frac{t}{2}.
\]

Summing up these values we obtain the lower bound:

\[
\frac{t}{2} \sum_{j=1}^{\frac{t}{2}} m_i \geq \frac{t}{2} \left( \frac{1}{2} - j + 1 \right)(k-1) = \frac{n}{4} \left( \frac{n}{2k} + 1 \right)(1 - \frac{1}{k}) = \Omega \left( \frac{n^2}{k} \right)
\]

Case 2. At least half of the used edges carried their first message to a processor \( a_i \). The case where minimum number of edges are used is the following:

1. \( a_i = i \).
2. All the above messages reached the processors \( a_1, a_2, \ldots, a_{\frac{t}{2}} \).

Since at least \( k-1 \) of the above messages reached each one of the above processors, and each such message implies usage of \( i \) edges (by its sender), we get that the number of used edges is at least

\[
\frac{t}{2} \sum_{i=1}^{\frac{t}{2}} (k-1) = \Omega \left( \frac{n^2}{k} \right).
\]

Since every algorithm for this problem is a global algorithm, its complexity is at least \( \Omega(n \log n) \) edges ([KMZ1]). This completes the proof:

Q.E.D.
The given lower bound is tight, since a simple algorithm can be designed that solves the problem using $O\left(\max\{n \log n, \frac{n^2}{k}\}\right)$ messages. The complexity of this problem, given a CR labeling is $\Theta(n)$, since any global algorithm requires $O(n)$ messages, and these messages suffice since the processors have a global sense of direction.

### 7.2. Restricted spanning tree

We now depart from the models defined before and deal with the *two-colors* model; we remain with a complete graph, but now the edges are partitioned into two sets, white edges and blue edges. Note, that this is not the usual meaning of edge coloring, i.e., many white or blue edges may be adjacent to the same vertex. Consider the following problem: given the two-colors model, construct a spanning tree that contains only white edges. We introduce this problem and model because in some cases its complexity is high, and in others, low. The complexity of this problem depends on the number of white edges adjacent to each vertex.

Denote *white degree* of a vertex as the number of white edges adjacent to it. When the white degree of all the vertices is less than $\frac{n}{2}$, one can view the white edges as weighted zero and the blue edges as weighted one, and derive tight lower and upper bounds of $\Theta(n^2)$ as they are derived for MST in [KMZ2, GHS]. Actually, if each vertex has to know whether such a tree exists, the bounds can be refined as follows:

**Corollary 4:** If the white degree of all the vertices is less than $\frac{n}{2}$, and $k$ is the number of white edges in the graph, then the edge complexity of constructing a white spanning tree is $\Theta(\max\{n \log n, k\})$.

Another case we deal with is when the white degree of all vertices is greater than or equal $\frac{n}{2}$. It is not important that the processors will have that knowledge before the algorithm starts because the cost of gathering this information is of $O(n \log n)$.

**Theorem 11:** If the white degree of all the vertices is greater than or equal $\frac{n}{2}$, then the edge complexity of constructing a white spanning tree is $\Theta(n \log n)$.

**Proof:** For the upper bound we show an algorithm:
Algorithm for white spanning tree

1. Elect a leader.

2. The leader chooses a vertex, and this vertex sends messages on all of its adjacent white edges. (The white used edges from this step on are the tree being constructed.)

3. Repeat until the white tree spans the graph:
   3.1. The leader chooses a vertex that does not belong to the tree.
   3.2. This vertex sends messages on unused white edges until the first message reaches a vertex that belongs to the tree.

Correctness of the algorithm: consider the graph induced by the white edges that are used since step (2). This graph is a forest where all of its elements except one are isolated vertices. At step (3), an isolated vertex is chosen, and this vertex “grows” a star shaped component, until it connects with the other component. This connection always takes place because in the graph induced by all of the white edges every two vertices have a common neighbor. Therefore, as step (3) is repeated, the component grows, until it includes the whole graph.

Complexity of the algorithm: step (1) costs $O(n \log n)$ messages ([Hu, KMZ1]). At steps (2) and (3) the number of messages sent is linear to the number of white edges used, hence only $O(n)$ messages. Thus, the message complexity of the algorithm is $O(n \log n)$.

Lower Bound: this model is weaker than the matching model, hence the complexity of every global algorithm is at least $\Omega(n \log n)$.

Q.E.D.

7.3: Chordal Ring

In [ASZ], chordal rings $C_n <a_1, a_2, \ldots, a_k>$, $1 < a_1 < a_2 < \cdots < a_k < n - 1$, in which $n$ processors are arranged on a ring and each processor knows the edges that leads to its neighbors on the ring and to processors in distances $a_1, a_2, \ldots, a_k$ clockwise. For the chordal ring $C_n <2, 3, \ldots, t>$ it is shown that if $t \geq \log n$ than leader election can be done using $O(n)$ messages. We extend this result to the case where the $i$-th chord leads to distance $2^i$.

Theorem 12: The message complexity of the election problem, given the chordal ring $C_n <2, 4, \ldots, 2^{\log(n-2)}>$, is $\Theta(n)$. 
Proof: First we prove for ring sizes that are powers of 2, i.e. \( n = 2^k \). We use the algorithm from [ASZ] that uses the chords to reduce the cost of sending a message to distance \( d \) on the ring, to at most \( \lceil \log d \rceil \). The number of active processors at the beginning of each phase is at most half of that number at the previous phase. (It follows from the fact that from each two neighboring active processors at the beginning of a phase, at most one will be active at the beginning of the next phase.) Consider the following execution \( e \) (of the algorithm on this chordal ring): at the beginning of phase \( p \), there are exactly \( \frac{n}{2^p} \) active processors, and the distances between two neighboring active processors are \( 2^p \). (Such execution exists as a result of applying the algorithm on a chordal ring with \( Q_n \) taken from [FL]s.) During a phase each processor sends one message to each of its active neighbors. Therefore, the cost of phase \( p \) is at most
\[
2 \times \frac{n}{2^p} \times \left\lceil \log(2^p) \right\rceil = 2^p \frac{n}{2^p} = 2^p \frac{n}{2^p},
\]
and the sum over all of the phases yields
\[
\sum_{p=0}^{\log n} 2^p \frac{n}{2^p} = O(n).
\]
Now we show that any other execution of the algorithm on a chordal ring of the same size does not use more messages. We compare the two executions by phases, that is, we show that the cost (i.e. number of messages sent during the phase) of phase \( p \) in execution \( e \) is not less than the cost of phase \( p \) in any other execution \( e' \). Denote \( D = \{d_i\}_{i=1}^{N_p} \) the distances in phase \( p \) in execution \( e \), and \( D' = \{d'_i\}_{i=1}^{N'_p} \) the distances in phase \( p \) in execution \( e' \). The cost of a phase is a function of the distances between neighboring active processors, and if \( D = \{d_i\}_{i=1}^{N_p} \) are these distances, then, the cost of the phase is at most \( \sum_{i=1}^{N_p} 2 \times \left\lceil \log d_i \right\rceil \). Note, that \( d_i = 2^p \text{ and } d_i' \geq 2^p \) for all \( i \), and
\[
\sum_{i=1}^{N_p} d_i = \sum_{i=1}^{N'_p} d_i' = n.
\]
Lemma 3: Given \( D \) and \( D' \), if \( d_i \geq 3 \), then
\[
\sum_{i=1}^{N_p} \log d_i \geq \sum_{i=1}^{N'_p} \log d_i'.
\]
Proof: Since the sum of the logarithms of a set of numbers equals the logarithm of their product, and \( \log(x) \)
is a monotonic increasing function, it suffices to show that
\[ \prod_{i=1}^{N'} d_i \geq \prod_{i=1}^{N'} d'_i. \]

For convenience, let \( v \) equal \( d_i \) (for all \( i \)). Then we can write \( D' = \{ q_1 v, q_2 v, \ldots, q_{N'} v \} \) where \( q_i \geq 1 \), and \( \sum_{i=1}^{N'} q_i = N \). Therefore we have
\[ \prod_{i=1}^{N'} d_i = v^N = v^{\sum q_i} = v q_1 v \times v q_2 v \times \cdots \times v q_{N'} v \] (1)
and
\[ \prod_{i=1}^{N'} d'_i = q_1 v \times q_2 v \times \cdots \times q_{N'} v. \] (2)

Consider the right hand sides of equations (1) and (2). It suffices to show that
\[ v^q \geq q_i v, \quad i = 1, \ldots, N'. \] (3)

Now, let's regard each side of inequality (3) as a function of \( q \) for fixed \( v \) and write \( f(q) = v^q \) and \( g(q) = v q \). From the nature of the linear \( f(q) \) and exponential \( g(q) \) functions, it follows that if for some point \( q_0 \)
\[ g(q_0) = f(q_0), \]
and
\[ g'(q_0) \geq f'(q_0) \]
then
\[ g(q) > f(q), \quad \forall q > q_0. \]

Now, in our case,
\[ g(1) = f(1) \]
\[ g'(1) = v \log_i v. \]
and
\[ f'(1) = v. \]

---

For convenience, we omit the index \( p \) from \( N \) and \( N' \) during the proof.
Hence, for \( v \geq 3 \)

\[
g'(1) > f'(1)
\]

and the Lemma follows.

Using the Lemma we bound the cost of each phase:

\[
\sum_{i=1}^{N_p} \left\lfloor \log d' \right\rfloor < N'_p + \sum_{i=1}^{N_p} \log d' \leq N'_p + \sum_{i=1}^{N_p} \log d_i.
\]

This we sum over the phases,

\[
\sum_{p=0}^{\log \left( \frac{n}{2^{p_r}} + 2p \frac{n}{2^{p_r}} \right)} = O(n)
\]

to get the required upper bound.

Now we extend the proof for chordal ring of any size. Let \( R \) and \( R' \) be chordal rings of sizes \( n \) and \( n' \), respectively, where \( n \) is any integer, and \( n' \) is the smallest power of 2 that is greater than \( n \). Given the ring \( R \), we show that any execution on it, has an execution on \( R' \) that uses at least as many messages. As is shown above, the complexity of the algorithm when applied on the ring \( R' \) is of \( O(n') \). Therefore the complexity of the algorithm is \( O(n) \) for all \( n \) (because \( n \) and \( n' \) are of the same order).

For each identity assignment of the ring there is a unique execution of the algorithm, because different delays on the messages does not alter the behavior of the algorithm. To be more exact: the arrangement of the identities of the active processors at the beginning of a phase dictates the behavior of the algorithm from this point and on. Thus, for a given assignment of identities for the ring \( R \) we construct an assignment for the ring \( R' \) that satisfies the following condition: Applying the algorithm on both rings, the identities arrangements of the active processors at the beginning of the second phase are equal.

We now show the construction. For convenience, we denote the processors around the ring \( P_1, P_2, \ldots, P_n \) and the identity of processor \( P_j \) as \( d_j \). Let \( id_1, id_2, \ldots, id_n \) be an assignment of identities to the ring \( R \) and without loss of generality, assume that \( id_1 \) is the smallest identity. (Simply start the numbering of the processors from the processor with the smallest identity.) First we augment the set of identities with \( n' - n \) more elements, \( id_{n+1}, \ldots, id_{n'} \) such that \( id_j < id_{j+1} \), for \( j = n+1, \ldots, n'-1 \), and \( id_n < id_1 \). The assignment for the ring \( R' \) is \( id_1, \ldots, id_n, id_{n+1}, \ldots, id_{n'} \).
We return to the discussion. Consider the first phase of the algorithm when it is applied on rings $R$ and $R'$. At $R'$ all the processors $p_{n+1}, \ldots, p_n$ become passive. Processor $p_1$ becomes passive in both rings. Processor $p_n$ receives on one of its sides the same message on both rings (i.e. from $p_{n-1}$), and on the other side receives a message with identity smaller then $id_n$ and $id_{n-1}$ on both rings. Hence, on both rings it acts the same (become passive or not). All the other processors receive exactly the same messages on both rings, and therefore act in the same way. Hence, the arrangements of the identities of the active processors at the beginning of the second phase in both rings are the same. The distance between two active processors $p_i$ and $p_j$ in $R'$ is greater or equal to that distance in $R$ and considering the first phase, we get that the execution on $R'$ uses more messages than the one on $R$. This completes the proof.

Q.E.D.

Acknowledgement: We thank Orman Gerstel for proposing the Matching model, and Alan Hartman for pointing out the reference [HCW].

REFERENCES

[ASZ] C. Attiya, N. Santoro and S. Zaks, *From rings to complete graphs - $\Theta(n\log n)$ to $\Theta(n)$ distributed leader election*, TR-446, Department of Computer Science, Technion, February 1987.


[Hu] P. A. Humblet, Selecting a leader in a clique in $O(n \log n)$ messages, MIT, internal memorandum.


