OPTIMAL ELECTIONS IN LABELED HYPERCUBES

Paola Flocchini and Bernard Mans

TR-231, DECEMBER 1993

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

Optimal Elections in Labeled Hypercubes*

Paola Flocchini†
(flocchin@scs.carleton.ca)

Bernard Mans‡§
(mans@scs.carleton.ca)

Abstract

We study the message complexity of the *Election* problem in Hypercube networks, when the processors have a "Sense of Direction", i.e. the capability to distinguish between adjacent communication links according to some globally consistent scheme.

We present an original $\Theta(N)$ messages algorithm which uses the natural dimensional labeling of the communication links; to date, the best existing bound was $O(N \log N)$ messages. This result answers a long-standing open problem posed by the $\Theta(N)$ messages bound for chordal rings [1].

We also consider the Election problem for Hypercube with another common labeling which is distance-based. We prove that, in this case the problem becomes drastically simplified and we present a $\Theta(N)$ messages solution. Finally, we study the communication cost to change one orientation labeling to the other and prove that $O(N)$ messages suffice.

1 Introduction

The problem of electing a leader is one of the most widely studied in the literature on distributed computing. The *Election* (or *Leader Finding*) problem consists to arrive from an initial configuration where all the processors are in the same state to a final configuration where exactly one processor is in a *leader* state and all the other processors are in state *lost*.

For an arbitrary network of N asynchronous processors and e bidirectional communication links, the Election problem has been proved to require $\Omega(e + N \log N)$ messages

*This work has been done while visiting School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6 Canada, phone: (613) 788-4333, Fax: (613) 788-4334

†Dipartimento di Scienze dell'informazione, Università di Milano, Via Comelico 39, 20135 Milano, Italy

‡support by Université du Québec à Hull, Case postale 1250, succ "B", Hull, Québec J8X 3X7 Canada

§supported previously by INRIA postdoc fellowship, Rocquencourt BP 105, 78153 Le Chesnay Cedex, France

(e.g., [12]), and such a bound is achievable [2]. If an *a priori* knowledge of the structure of the system is available to the processors (i.e. topological awareness), it is possible to exploit topological properties to reduce the message complexity of any Election algorithm to the smallest of two factors: the number of links e and the $(N \log N)$ factor. In particular, $\Theta(N \log N)$ bounds have been proved for the complete network [7, 8] in such a case. Obviously, such an approach does not yield any improvement in a topology where the number of edges e is of the same order than $N \log N$. This is the case of the Hypercube, and an $O(N \log N)$ message complexity can be trivially achieved. An open problem of finding an $o(N \log N)$ algorithm for the Hypercube has remained open to date.

Another crucial factor, called **Sense of Direction**, has been identified to be significant for the computability and the communication complexity of Elections problems [13]. The Sense of Direction refers to the capability of a processor to distinguish between adjacent communication lines, according to some globally consistent scheme.

For instance, in a ring network this property is usually referred to as orientation, which expresses the processor's ability to distinguish between *left* and *right*, where *left* means the same to all processors. Sense of Direction in a complete network is the knowledge of some predefined Hamiltonian cycle and the existence of a label on each link at each processor, that represents the distance along this cycle to the processor at the other end of the link. A similar definition exists for chordal rings or circulant graphs. In general, Sense of Direction can be defined by fixing a cyclic ordering of all the processors and labeling each incident link according to the distance in the above cycle to the other node reached by this link.

The availability of this **distance Sense of Direction** has been shown to have some positive impact on the message complexity. For instance, in arbitrary networks, the Election problem can be solved using $\Theta(N \log N)$ messages if Sense of Direction is available [10]. For a complete graph, the Election problem can be solved using $O(N)$ messages if Sense of Direction is available [9] ; such an improvement can be achieved even if each node has Sense of Direction on just $O(\log N)$ of its incident arcs [1]. Similar results hold for circulant graphs or chordal rings with $O(\log N)$ incident arcs [1].

The important open question is whether or not the Hypercube network, which has $O(\log N)$ arcs but is not a chordal ring, can support an Election Algorithm with $O(N)$ messages when the Sense of Direction is available. The question is all the more relevant since $O(N)$ messages suffice for the chordal ring $\langle 1, 2, 2^2, \cdots, 2^{\lceil \log N \rceil} \rangle$ [1], which closely resembles an Hypercube.

This paper answers this long-standing open problem in the affirmative by presenting an original $\Theta(N)$ messages algorithm for the Election problem with the Sense of Direction. Such an algorithm is given with the natural labeling of link based on the dimensional structure of Hypercube as defined in [14].

Because of the difference that exists between the common distance labeling and the dimensional one for the Hypercube, denoted **matching Sense of Direction** as defined in [5], we study the Election problem in the Hypercube with a distance Sense of Direction.

We show that in this case the problem becomes trivial, mostly thanks to a surprising graph property introduced by the distance Sense of Direction.

To complete the understanding of the impact of the Sense of Direction on the Hypercube network, we study also the complexity relationship between these two models of "Sense of Direction" by presenting an $O(N)$ message complexity algorithm to change from one Sense of Direction to another. In the concluding remarks, we derive a $\Theta(N)$ messages algorithm for the *Wake-up* problem in a Hypercube network. We show the efficiency of the presented solutions by comparison with a synchronous implementation.

After introducing several notations and preliminaries in Section 2, we present the algorithms for the Election problem for the matching model in Section 3, and for the distance model in Section 4. In Section 5, we present the $O(N)$ messages algorithm to change one orientation labeling to the other. Section 6 includes immediate extensions and related works on these results.

2 Preliminaries

An n-dimensional *hypercube* network is represented by an undirected graph consisting of $N = 2^n$ vertices which can be labeled from 0 to $2^n - 1$ such that there is an edge between any two vertices if and only if the binary representation of their labels differs by one and only one bit. The n-dimensional hypercube has $\frac{1}{2}nN$ edges, i.e. $\frac{1}{2}N \log N$, and every node has degree n. Each edge between two nodes is labeled on each node by the dimension of the bit of the identity in which they differ.

The computation model that we consider is a network of $N = 2^n$ asynchronous processors connected in a Hypercube-like topology. Every processor P_i in the Hypercube network has a distinct value id_i chosen from some infinite totally ordered set ID, and is aware only of its own value (does not know the identities of its neighbours). Every processor performs the same algorithm. We assume that the messages on each arc arrive with no error, in a finite but unbounded delay and in a FIFO order. The complexity measure is the maximum number of messages sent during any possible execution.

If the intuitive labeling based on the dimension is available, the network is called to have the property denoted as **Sense of Direction**. The dimensional labeling guarantees a strong consistency [14]: both incident nodes of the edge give the same label to this link. This *dimensional* Sense of Direction, (a) in figure 1, will be denoted by the **matching model** as defined in [5].

The other model of Sense of Direction, commonly used for other topologies, [1, 5, 9, 10] is denoted by the **distance model**. In particular, if a link, between two processors p and q, is labeled by distance d at processor p, this link is labeled by $N - d$ at the other incident processor q, where N is the number of processors. This distance Sense of Direction can be provided in the Hypercube, as shown in figure 1 (b).

In this latter case, a Hamiltonian cycle is the natural way to fix an ordering of all the nodes. Several different Hamiltonian cycles can be built in a Hypercube, but the one

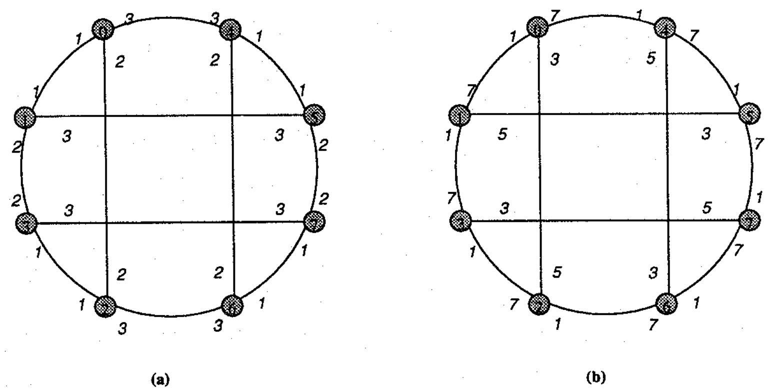


Figure 1: Model of Sense of Direction for Hypercube: (a) Matching, (b) Distance

based on the *binary reflected* Gray code [3, 4] will be heavily used in the sequel. Ignoring the starting vertex identity, a handy notation of a Hamiltonian cycle in a Hypercube corresponds to the list of the labels of the links traversed during the construction. Such a N-tuple is classically called a *coordinate sequence* for the cycle. An obvious example for a reflected coordinate cycle with $n = 4$ in the matching model is

$$\langle 1213121412131214 \rangle$$

In order to follow a Hamiltonian cycle, a simple message which contains the rank in the cycle of the last processor visited is passed and increased, e.g. the first processor has to send the message through the link labeled 1 and the $4th$ processor which receives the message knows that it has to send it through the link labeled 3. In fact, a Hamiltonian cycle in a n-dimensional hypercube can be built by induction on a $(n-1)$-dimensional hypercube: the new coordinate sequence A_n is obtained from the previous one B_{n-1} and the concatenation of the new value of degree n, i.e. $A_n = \langle B_{n-1} n B_{n-1} n \rangle$. Therefore, the coordinate cycle can be computed locally from scratch.

3 Election Algorithm with the Matching Model

In this section, we present an asynchronous distributed algorithm for the Election on the Hypercube with a $\Theta(N)$ message complexity with a matching Sense of Direction. We emphasize the fact that this algorithm is original and does not built a spanning tree.

The Election process may be independently started by any subset of the processors. Let notice that the processors do not have (or do not know) the binary node labeling from 0 to $2^n - 1$ which is usually assumed for Hypercube (otherwise this provides a trivial solution to the problem).

Informal description The algorithm proceeds in phases of a tournament. Initially, each node of the network is a *Duellist*. At the end, all the nodes are *Seconds* except one which is still a Duellist (i.e. the *leader*). In every phase of the algorithm, each successful Duellist goes for the next duel by challenging another Duellist. The algorithm is done by repeated sequence of a duel which corresponds to a combination of two cubes into a larger one, and, thus, takes $\log N$ steps (namely one per dimension).

The basic idea is that, at each step k, each Duellist has to challenge (send an attack) and to be challenged (receive an attack) by his respective Duellist in the rank of the tournament (the Duellist node in his cube image regarding the dimension k). This makes the two opposite Duellists hand-shake: the Duellist with higher id who receives an attack from Duellist with lower id, wins the duel of step k and becomes Duellist of level $k+1$; conversely, the Duellist with lower id who receives an attack from Duellist with higher id looses, becomes *Second* of level k and keeps the path between himself and the winner. Neither acknowledgment nor surrender messages are required. The task of a Duellist is to fight a duel, whereas the task of a Second is to forward an incoming attack to his Duellist.

The fundamental property used in the algorithm is the property that, at any step i, a Second of level k (with $k < i$) knows the *coordinate sequence* traversed by his Duellist (of level $k+1$), i.e. the Duellist against whom he lost. Thus, the position (regarding the dimensional direction) of a Duellist of level $k+1$ is unambiguously known by the Second, and a shortest path between them can be computed on the fly by "compressing" (see below) the coordinate sequence received during the fatal attack. We denote with $Duellist(s)$ and $Second(s)$ entities who, respectively, won and lost the duel of step s.

When an attack from $Duellist(s)$ reaches a $Second(k)$ (with $k < s$), the Second forwards it to the node lately known as the $Duellist(k+1)$. If this node is not any more a Duellist (namely has been defeated and became Second), it will forward it to its respective $Duellist(k+2)$. The process is repeated until the $Duellist(s)$ is found. This local view of a Second is exploited by forwarding an attack from Duellist(s) that reaches a Second(k) (with $k < s$) through the path $\langle$ Second(k) Second($k+1$) $\ldots$ Second($s-1$) Duellist(s)$\rangle$.

To reduce the amount of communication, an attack from $Duellist(s)$ reaching a $Duellist(k)$ (with $k < s$) is delayed until the $Duellist(k)$ either reachs the required level s, or becomes a Second $Second(k)$ able to forward the message to its respective $Duellist(k+1)$.

At the beginning all nodes are $Duellist(0)$. The algorithm terminates after the $(\log N)$th step with a unique Duellist $Duellist(\log N)$.

Algorithm

Local information used:

(i) $State_p : \{Duellist, Second, Leader\}$ is the state of a node (initially *Duellist*).
(ii) For each node p, Id_p is its identity, and $level_p$ is its level,
(iii) For each Second p, $NextDuellist$ denotes the path (the coordinate sequence) from the Second p at level $level_p$ to the Duellist of level $level_p + 1$.

Messages Used:
(i) ATTACK: This message represents the challenge for a duel and contains: $(Id, lev,$-$source, dest)$, where *Id* is the identity of the initiator of the attack, *lev* is the step of the attack, *source* is the path (the coordinate sequence) from the attacking Duellist to the present node receiving the attack. This list will be stored to know the followed path. Finally, *dest* is the path (the coordinate sequence) from the present owner of the message to a node target. This list is used to forward an attack by the shortest path between a Second to his Duellist.
(ii) LEADER: it is broadcasted by the unique remaining Duellist (the Leader) to terminate the execution of the algorithm and inform of his Id (*LeaderId*).

Any number of processors can spontaneously start the execution of the algorithm; this is modeled by the reception of a WAKEUP message.

Functions Used:
(i) first(list) gets the rank of the first non-zero bit of a list *list* of $\log N$ bits.
(ii) list $\oplus$ i (resp. $\ominus$) updates the given path by adding (resp. eliminating) a label i in a coordinate sequence *list*. Then, the compression is done by eliminating every pair of coordinates (e.g. a sequence $\langle 123142 \rangle$ is equivalent to $\langle 34 \rangle$). This guarantees that the coordinate sequence will never exceed $\log N$ labels and then can be represent with $\log N$ bits: a bit of rank i set to 1 means that the label i is present in the coordinate sequence. Namely, the operations $\oplus$ and $\ominus$ are identical: when applied on bit i of value b_i, they both set its value to $(1 - b_i)$.

procedure *Election(p)*
begin
/* initially - processor can be asleeped */
(0) Upon RECEIPT of *(WAKEUP)* or any other message on any arc
    if message $\neq$ WAKEUP **then** delay message
    $State_p$:= *Duellist* ; $level_p := 0$
    $Id := Id_p$; $lev := 0$; $source := []$; $dest := []$
    SEND *ATTACK(Id,lev,source,dest)* on arc labeled 1
end WAKEUP

• If $State_p = Duellist$:

(1) Upon RECEIPT of *ATTACK(Id,lev,source,dest)*
  with $(lev = level_p)$ on arc labeled r
    if $(Id_p > Id$ and $lev = n)$ **then**
      $State_p$:= *Leader*
      SEND(LEADER,Id_p) on all arcs
    endif
    if $(Id_p > Id$ and $lev < n)$ **then**
      $level_p := level_p + 1$
      $Id := Id_p$; $lev := lev + 1$; $source := []$; $dest := []$
      SEND(ATTACK(Id,lev,source,dest)) on arc labeled $r + 1$
      accept delayed message with $lev = level_p$ if arrived
    endif

```
        if (Id_p < Id) then
           State_p := Second
           NextDuellist := source ⊕ r
           accept all delayed messages if any
        endif
end ATTACK

(2) Upon RECEIPT of ATTACK(Id,lev,source,dest)
    with (lev > level_p) on arc labeled r
           delay message
end ATTACK
```

• If $State_p = Second$:

```
(3) Upon RECEIPT of ATTACK(Id,lev,source,dest)
    with (dest ≠ []) on arc labeled r
        l := first(dest) ; dest := dest ⊖ l ; source := source ⊕ r
        SEND(ATTACK(Id,lev,source,dest)) on arc labeled l
end ATTACK

(4) Upon RECEIPT of ATTACK(Id,lev,source,dest)
    with (dest = []) on arc labeled r
        dest := NextDuellist
        l := first(dest) ; dest := dest ⊖ l ; source := source ⊕ r
        SEND(ATTACK(Id,lev,source,dest) on arc labeled l
end ATTACK

(5) Upon RECEIPT of LEADER(LeaderId) on arc labeled r
    Forall k < r
        SEND(LEADER(LeaderId)) on arc labeled k

end LEADER

end Election(p)
```

Step analysis Let $d(k-1,k)$ be the maximum distance between Duellist($k-1$) and Duellist(k), clearly $d(k-1,k) = k$. Let $F(i)$ be the maximum number of links that have been traversed at step i to go from Duellist(0) to Duellist($i-1$).

Obviously, in Step 1 the N Duellist(0) attack their corresponding Duellists(0), and, thus, each has to traverse only one link: $F(1) = 1$. In Step 2, the $\frac{N}{2}$ Duellists(1) send an ATTACK; the attack can reach a Second(0) who has to forward it to Duellist(1), thus the maximum number of links to be traversed is $F(2) = 1 + d(0,1) = 1 + 1 = 2$. In Step 3, the $\frac{N}{2^2}$ Duellist(2) send an ATTACK; the attack can reach a Second(0) who forwards it to the Duellist(2) through a Second(1), thus $F(3) = 1 + d(0,1) + d(1,2) = 4$. For any step i, the $\frac{N}{2^{i-1}}$ Duellist(i-1) start an ATTACK; the attack, in the worst case, has to follow the path: $\langle$Second(0) ... Second($i-2$) Duellist($i-1$)$\rangle$, the number of links to be traversed is

$$F(i) = 1 + \sum_{k=1}^{i-1} d(k-1,k) = 1 + \sum_{k=1}^{i-1} k = 1 + \frac{i \cdot (i-1)}{2}$$

Full analysis

Theorem 3.1 *The total number of messages sent during the algorithm for the Election problem is at most*

$$7N - (\log N)^2 - 3\log N - 7$$

Proof At any step i at most $\frac{N}{2^{i-1}}$ nodes send an ATTACK message. During each of the $\log N$ steps at most $F(i)$ forwarding messages are required to reach the target Duellist.

Thus, the total number M of messages is

$$M = \sum_{i=1}^{logN} \frac{N}{2^{i-1}} \cdot F(i) = \sum_{i=1}^{logN} \frac{N}{2^{i-1}} \cdot \left(1 + \frac{i \cdot (i-1)}{2}\right)$$

$$= N \left(\sum_{i=1}^{logN} \frac{i^2}{2^i} - \sum_{i=1}^{logN} \frac{i}{2^i} + 2 \sum_{i=1}^{logN} \frac{1}{2^i} \right)$$

Since

$$\sum_{i=1}^{logN} \frac{i^2}{2^i} = \frac{6N - 4\log N - (\log N)^2 - 6}{N} \quad (1)$$

$$\sum_{i=1}^{logN} \frac{i}{2^i} = \frac{2N - \log N - 2}{N} \quad (2)$$

$$\sum_{i=1}^{logN} \frac{1}{2^i} = \frac{N-1}{N} \quad (3)$$

and, since exactly $(N-1)$ LEADER messages are required to broadcast the termination and leader identity, we prove the theorem. □

Theorem 3.2 *Each message is composed of at most* $(\log M + \log\log N + 2 \cdot \log N)$ *bits, where the identity of a node is at most* M.

Proof Each ATTACK message contains the identity Id_p of the attacking Duellist, the level lev of the attack whose value is at most $\log N$, the location of the attacking Duellist ($\log N$ bits) and the location of the attacked Duellist ($\log N$ bits). A LEADER message contains only an identity. □

Correctness Several properties are introduced for the correctness. In the following, numbers between parentheses refer to corresponding sections of the algorithm. We shall denote by Duellist(i) (respectively Second(i)) a Duellist (respectively a Second) at level i.

Lemma 3.1 *Let an ATTACK message contains a level lev initiated by a Duellist p of level* $level_p$*, then*
(i) the message has been originated through a link labeled lev such that $lev = level_p$*,*
(ii) after traversing the link through which it has been originated, the message can not traverse a link labeled d, with $lev < d$.

Proof (i) immediate from (1). (ii) by induction: since a Second never modifies the level of an ATTACK message (3) (4), and a Second never initiates a duel, the ATTACK traverses only links of smaller level. Therefore the ATTACK message can not exit the subcube of degree *lev* whose it belongs. □

The next lemma guarantees that a Second knows the shortest path to reach its Duellist.

Lemma 3.2 *A Second(i) knows the shortest path (a coordinate sequence with minimum length) to reach its respective Duellist(i + 1).*

Proof The correctness of the path is guaranteed through the compression of the *source* list which stores the labels traversed links (3) and (4). By contradiction; assume that the length of the path is strictly greater than the shortest one, this would imply that at least two labels with the same direction exist in the coordinate path, which is forbidden by the *compress* mechanisms $\oplus$ and $\ominus$. □

We show now that no infinite delays are introduced during the execution of the algorithm.

Lemma 3.3 *A deadlock may not be introduced by the waiting that arises when some nodes must wait until some condition holds.*

Proof Since the message broadcasted by the LEADER is forwarded immediately upon reception, only ATTACK messages may create a cycle. In Lemma 3.1 the partitioning of the subcube traversed by a message ATTACK has been shown and, thus, only cycle between two Duellists of the same level can be created. The only situation in which an entity is waiting for an event is when a Duellist a waits to be accepted by another Duellist b with $level_b < level_a$ to reach the respective level $level_a$, (2). By induction, the extreme case occurs when a chain of waiting processor such that $level_x < .. < level_b < level_a$ is created. Thus, by (0), the total ordering of the chain forbids the creation of a cycle in such a chain.

□

We now prove that an ATTACK message always reaches the target Duellist.

Lemma 3.4 *An ATTACK from a Duellist(i) ($i < \log N$) will eventually reach another Duellist(i).*

Proof The ATTACK is sent by a Duellist(i) through link $i+1$ (Lemma 3.1 (i)). Since this attack will remain inside the $(i+1)$-cube (Lemma 3.1 (ii)), it will reach either (a) a Duellist(i') (with $i' \leq i$) or (b) a Second(i'), (with $i' < i$).

In case (a), if $i' = i$ the lemma is proved. Otherwise the message is delayed until the Duellist(i') receives an ATTACK of level i'. This delaying entity is waiting for an ATTACK from his image Duellist(i') to whom he has already sent his ATTACK. This latter attack, might have reached a Duellist(i'') (with $i'' < i'$) which must wait to reach level i' before fighting. The recursive argument can be repeated and might create a chain of delaying entities. The total ordering of the chain forbids the creation of an infinite chain. Namely, this process ends when a Duellist can reply to the attack (in the worst case when Duellist(0) is reached) and reactivate successively the waiting entities.

Note that this argument proves also that a Duellist(i) ($i < \log N$) will eventually become either a Second(i), or a Duellist($i + 1$).

In case (b) two situations may occur depending on whether the *destination* is known or not (i.e., empty). If not, (4), the Second(i') sets the *destination* with the shortest path to reach Duellist($i' + 1$) and then forwards the attack. If the ATTACK message contains a non empty *destination* the correctness of such a traversed path is guaranteed by Lemma 3.2. □

The next lemma shows that there is exactly one Duellist at level i in each i-cube.

Lemma 3.5 *A Duellist(i) is the unique Duellist of an undirected i-cube in which there exist* $2^i - 1$ *Seconds with a level strictly less than i.*

Proof By induction. Each Duellist(0) is a 0-cube. Let the theorem hold for i and consider the fights of level $i + 1$. Each Duellist(i) sends an attack through the link labeled $i + 1$. From lemma 3.1 (ii) the attack remains inside the ($i + 1$)-cube in which there exists a unique image Duellist (by inductive hypothesis). From lemma 3.4, the attack will reach this Duellist and only one of the two will become Duellist($i + 1$). □

By Lemmas 3.3, 3.4 and 3.5, it follows that

Theorem 3.3 *The algorithm terminates correctly after* $\log N$ *steps and elects a unique* leader.

4 Election Algorithm with the Distance Model

We present a distributed algorithm for the Election problem on the Hypercube with a $\Theta(N)$ message complexity with a distance Sense of Direction. We first prove that the symmetry is broken by this model of Sense of Direction and, thus, that the leader Election does not require a global maximum-finding algorithm.

Theorem 4.1 *For any oriented hypercube with the distance Sense of Direction obtained by the Gray reflected code, exactly four processors have a link labeled* $\frac{N}{2} - 1$.

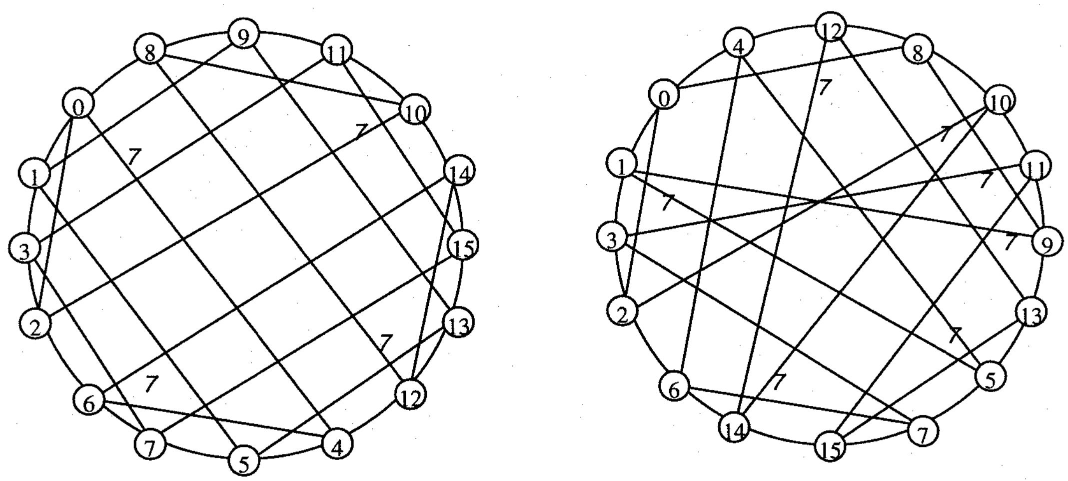
Proof By induction, assuming that all the Hamiltonian cycle that has been built in the Hypercube respect the binary reflected code construction. Without loss of generality of the proof, we assume in the sequel that all the processors have a number in $[0..(N-1)]$ which depends of their respective position in the Hamiltonian cycle.

The Theorem is true for $N = 8$, see figure 1(b). By construction, the Hamiltonian cycle for the Hypercube with the successive degree is obtained by taking this smaller hypercube as the first half-cycle, add the mirror image as the second half-cycle, and finally join each processor's image by a link.

In the new subset of links obtained by the mirror process (in between the two half-cycles), only the two processors and their respectives images assigned to the first quarter of the cycle have link with a $\frac{N}{2}-1$ distance, thus, only two processors have such a labeled link (the respective images have a label with value $\frac{N}{2}+1$): processor numbered $\frac{N}{4}$ and processor numbered $3\frac{N}{4}$.

In each of the two half-cycles of size $\frac{N}{2}$, by definition, only one link may have a $\frac{N}{2}-1$ distance, since this is the maximum distance in the half cycle. This link always exists since it is the one between the first and the last processor in the half-cycle which are joined (the terminal link of the smaller Hamiltonian cycle). Thus, only two processors have such a labeled link: processor numbered 0 and processor numbered $\frac{N}{2}$.

The four processors are in the respective cycle positions: $0, \frac{N}{4}, \frac{N}{2}, 3\frac{N}{4}$ (figure 2). □



(a) (b)

Figure 2: Hamiltonian Cycle Impact on Distance: (a) Reflected, (b) Arbitrary

remark The Gray reflected code is a necessary condition since some other Hamiltonian cycles can be followed in the hypercube providing an arbitrary number of nodes with an edge labeled by a $\frac{N}{2}-1$ distance. For instance, the $\langle 1213414243212343 \rangle$ coordinate cycle, initialized in processor 0 in a 4-hypercube (figure 2(b)), provides seven (7) such a node (nodes with hypercube identities {1,14,5,9,11,10,12}).

Corollary 4.1 *In the case of an oriented Hypercube with the distance Sense of Direction built on the reflected code, the overall message cost required in order that every nodes detect if it is elected or not, is* $O(1)$.

Proof The four processors with labels $\frac{N}{2}-1$ and their immediate predecessors (the four processors with a link labeled $\frac{N}{2}+1$) built a cycle of 8 processors. An algorithm using only this cycle elects a leader in between them in $O(1)$ messages. All the other nodes already know that are not elected by checking their link labels. □

Corollary 4.2 *An algorithm with an* $\Theta(N)$ *message complexity for the Election problem can be given for any oriented Hypercube with the distance Sense of Direction built on the reflected code.*

Proof A broadcast phase which uses exactly N messages is initiated by the Leader and terminates the algorithm by informing all the other processors of his identity. Each processor receiving the message on link $(N-1)$ forwards it on link 1. Clearly, the required $\Omega(N)$ bound is reached and, thus, the $\Theta(N)$ message complexity is proved. □

5 From one Sense of Direction to Another

Without a labeling of the links, the message complexity required to build the orientation of hypercubes has been proved [14] to be at least $\frac{1}{2}N(\log N-1)$.

We show that the distributed complexity of the orientation labeling decreases if the network is already oriented by another model of sense of direction. We present two $O(N)$ algorithms to re-label the links from one model of Sense of Direction to another.

Theorem 5.1 *Any oriented Hypercube with the matching Sense of Direction can be re-oriented with the distance Sense of Direction in* $O(N)$ *messages.*

Proof The algorithm executes three phases. The first phase elects a unique leader with the algorithm presented in Section 3.

In the second phase, the Hamiltonian cycle (initiated by the Leader) is built using the binary reflected Gray-code as described in Section 2. During this phase, each processor learns its relative number *cid* on the cycling path (*cid* of initiator is zero) and stores it. The incoming and outgoing edges respect their order in the coordinate sequence. These links are relabeled by $N-1$ for the one with the incoming message and by 1 for the other with the outgoing message.

The third phase consists to relabeled the $n-2$ untouched links, but is locally computated. On each node *cid*, a link previously labeled with the dimension d has to be relabeled by the distance:

$$(2^d-1-2\cdot(cid \bmod 2^d)) \bmod N$$

This result is fully based on the fact that the Hypercube, by construction of the Hamiltonian cycle, is split (for each d) in a set of equal slices of size 2^d. Each link labeled d joins two reflected images, and, thus, the distance on the cycle between the two adjacent nodes is twice the distance from this node to the end of the cube of size 2^d in the cycle.

Clearly, the two first phases take $O(N)$ messages respectively, and, thus, we prove the Theorem. □

Theorem 5.2 *Any oriented Hypercube with the distance Sense of Direction can be re-oriented with the matching Sense of Direction in $O(N)$ messages.*

Proof Similar than Theorem 5.1. The algorithm executes three phases. The first phase selects an initiator as shown in Corollary 4.1 using $O(1)$ messages.

In the second phase, since a Hamiltonian cycle built using the binary reflected Gray code is known following the path on outgoing link labeled 1 (the coordinate cycle is a sequence of N one), a counting message is passed through this cycle, so that each processor is able to learn its relative number *cid*. Upon receipt of this message, each processor re-labels its incoming link (previously $N-1$) and outgoing link (previously 1) regarding its rank on the A_n coordinate sequence described in Section 2.

The final phase is also a local computation which reverses the process used in Theorem 5.1. On each node, an edge previously labeled *dist* has to be re-labeled by the dimension:

$$\lceil \log((dist + 2 \cdot cid) \bmod N) \rceil$$

Only the second phase takes $O(N)$ messages, whereas the first takes only $O(1)$, and, thus, we prove the Theorem. □

6 Concluding Remarks and Related Works

The efficiency of the algorithms presented is also emphasized by the fact that each of them can be executed with a synchronous model. No main modification is required. The execution reached the optimal $O(N)$ message complexity and the $O(\log N)$ time complexity. The execution of the algorithm with the distance model is identical, whereas execution with the matching model corresponds to the simpler case where there is no delay (i.e. no challenge from a Duellist with higher level).

We derive a $\Theta(N)$ messages algorithm for the distributed problems of *Wake-up*. The *Wake-up* concerns the problem where some awakened processors must wake up the entire network in the absence of global start-up. The goal is to minimize the amount of communication used since the algorithm may be independently started by any arbitrary subset of the processors (see [6] for more details). The algorithm used for the Election problem in the matching model is the same, without the final broadcast. For the distance model, each awakened node sends a one bit message through link labeled 1 (to his neighbor on the cycle $\langle 1^N \rangle$) who will forward it if he was not already awake. These algorithms take respectively $O(N)$ messages.

After completing this paper, the authors learned that an election algorithm with the matching labeling has been obtained independently by Tel [15]. The algorithm of Tel is based on a match-making technique from Mullender and Vitanyi [11] and uses slightly more messages than our. Both algorithms terminates in $O(N)$ time in the worst-case (the time complexity can not exceed the message complexity). However, the size of the message in our algorithm has $2 \cdot \log N$ more bits.

Acknowledgements We would like to thank Nicola Santoro for his helpful comments and for bringing this problem to our attention.

References

[1] H. Attiya, J. van Leeuwen, N. Santoro, and S. Zaks. Efficient elections in chordal ring networks. *Algorithmica*, 4:437–446, 1989.

[2] R.G. Gallager, P.A. Humblet, and P.M. Spira. A distributed algorithm for minimum spanning tree. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, 1983.

[3] E.N. Gilbert. Gray codes and paths on the n-cube. *Bell Systems Technical Journal*, 37:815–826, 1958.

[4] F. Harary, J.P. Hayes, and H.-J. Wu. A survey of the theory of hypercubes graphs. *Comput. Math. Applic.*, 15(4):277–289, 1988.

[5] Z. Harpaz and S. Zaks. Sense of direction in complete distributed networks. Technical Report TR-453, Department of Computer Science, Technion, Haifa 32000, Israel, April 1987. extended abstract in 23rd Allerton Conference on Communication, Control and Computing, October 1985.

[6] A. Israeli, E. Kranakis, D. Krizanc, and N. Santoro. Time-message trade-offs for the weak unison problem. Technical Report TR-93-222, School of Computer Science, Carleton University, Canada, 1993. To appear in Proc. of CIAC'94, Rome, 1994.

[7] E. Korach, S. Moran, and S. Zaks. Tight lower and upper bounds for a class of distributed algorithms for a complete network of processors. In *Proceedings of 3rd Symposium on Principles of Distributed Computing (PODC'84)*, pages 199–207, Vancouver, Canada, August 1984.

[8] E. Korach, S. Moran, and S. Zaks. Optimal lower bounds for some distributed algorithms for a complete network of processors. *Theoretical Computer Science*, 64:125–132, 1989.

[9] M.C. Loui, T.A. Matsushita, and D.B. West. Election in complete networks with a sense of direction. *Information Processing Letters*, 22:185–187, 1986. see also *Information Processing Letters*, 28:327, 1988.

[10] B. Mans and N. Santoro. On the impact of sense of direction in arbitrary networks. Technical Report TR-93-232, School of Computer Science, Carleton University, Canada, 1993.

[11] S.J. Mullender and P.M. Vitanyi. Distributed match-making. *Algorithmica*, 3:367–391, 1988.

[12] N. Santoro. On the message complexity of distributed problems. *Journal of Computing Information Science*, 13:131–147, 1984.

[13] N. Santoro. Sense of direction, topological awareness and communication complexity. *SIGACT NEWS*, 2(16):50–56, summer 1984.

[14] G. Tel. *Lectures on Parallel Computation*, volume 4 of *Cambridge International Series on Parallel Computation*, chapter 9: Network Orientation, Ed. A. Gibbons and P. Spirakis. 1993.

[15] G. Tel. Linear election for oriented hypercube. Technical Report TR-RUU-CS-93-39, Utrecht University, Department of Computer Science, The Netherlands, 1993.

School of Computer Science, Carleton University
Recent Technical Reports

TR-211 **Generating Triangulations at Random**
Peter Epstein and Jörg-Rüdiger Sack, August 1992

TR-212 **Algorithms for Asymptotically Optimal Contained Rectangles and Triangles**
Evangelos Kranakis and Emran Rafique, September 1992

TR-213 **Parallel Algorithms for Rectilinear Link Distance Problems**
Andrzej Lingas, Anil Maheshwari and Jörg-Rüdiger Sack, September 1992

TR-214 **Camera Placement in Integer Lattices**
Evangelos Kranakis and Michel Pocchiola, October 1992

TR-215 **Labeled Versus Unlabeled Distributed Cayley Networks**
Evangelos Kranakis and Danny Krizanc, November 1992

TR-216 **Scalable Parallel Geometric Algorithms for Coarse Grained Multicomputers**
Frank Dehne, Andreas Fabri and Andrew Rau-Chaplin, November 1992

TR-217 **Indexing on Spherical Surfaces Using Semi-Quadcodes**
Ekow J. Otoo and Hongwen Zhu, December 1992

TR-218 **A Time-Randomness Tradeoff for Selection in Parallel**
Danny Krizanc, February 1993

TR-219 **Three Algorithms for Selection on the Reconfigurable Mesh**
Dipak Pravin Doctor and Danny Krizanc, February 1993

TR-220 **On Multi-label Linear Interval Routing Schemes**
Evangelos Kranakis, Danny Krizanc, and S.S. Ravi, March 1993

TR-221 **Note on Systems of Polynomial Equations over Finite Fields**
Vincenzo Acciaro, March 1993

TR-222 **Time-Message Trade-Offs for the Weak Unison Problem**
Amos Israeli, Evangelos Kranakis, Danny Krizanc and Nicola Santoro, March 1993

TR-223 **Anonymous Wireless Rings**
Krzysztof Diks, Evangelos Kranakis, Adam Malinowski, and Andrzej Pelc, April 1993

TR-224 **A consistent model for noisy channels permitting arbitrarily distributed substitutions, insertions and deletions**
B.J. Oommen and R.L. Kashyap, June 1993

TR-225 **Mixture Decomposition for Distributions from the Exponential Family Using a Generalized Method of Moments**
S.T. Sum and B.J. Oommen, June 1993

TR-226 **Switching Models for Non-Stationary Random Environments**
B. John Oommen and Hassan Masum, July 1993

TR-227 **The Probability of Generating Some Common Families of Finite Groups**
Vincenzo Acciaro, September 1993

TR-228 **Power Roots of Polynomials over Arbitrary Fields**
Vincenzo Acciaro, September 1993

TR-229 **Optimal Parallel Algorithms for Direct Dominance Problems**
Amitava Datta, Anil Maheshwari and Jörg-Rüdiger Sack, October 1993

TR-230 **Uniform Generation of Forests of Restricted Height**
M.D. Atkinson and J.-R. Sack, October 1993

TR-231 **Optimal Elections in Labeled Hypercubes**
Paola Flocchini and Bernard Mans, December 1993