Linear Election for Oriented Hypercubes

G. Tel

RUU-CS-93-39
December 1993
Linear Election for Oriented Hypercubes

G. Tel

Technical Report RUU-CS-93-39
December 1993

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
Linear Election for Oriented Hypercubes

Gerard Tel

Department of Computer Science, University of Utrecht,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
E-mail: gerard@cs.ru.nl

Abstract

In this article we propose an election algorithm for the oriented hypercube, where each edge is assumed to be labeled with its dimension in the hypercube. The algorithm exchanges $O(N)$ messages and uses $O(\log^2 N)$ time (where $N$ is the size of the cube).

A randomized version of the algorithm achieves the same (expected) message and time bounds, but uses messages of only $O(\log \log N)$ bits and can be used in anonymous hypercubes.

1 Introduction

The election problem is one of the most intensively studied problems in distributed algorithms research. In addition to its practical importance, the problem has developed to a "benchmark" to study the complexity effects of different model assumptions. In one line of this research, initiated by Santoro [San84], it was investigated how knowledge of topology or orientation influences the complexity of the election problem. (In the following discussion we only consider message complexity, and $N$ and $E$ denote the number of processes and links in the network.) For networks of arbitrary, unknown topology an $O(N \log N + E)$ algorithm was given by Gallager et al. [GHS83], and this is also a lower bound.

The existence of an orientation does not help in two important classes of networks, namely rings and tori. In unoriented rings, Franklin's algorithm [Fra80] uses $O(N \log N)$ messages, while the $O(N \log N)$ lower bound [Bur56, Bod91] applies to oriented rings as well. Peterson [Pet88] proposed an $O(N)$ algorithm for election in unoriented tori (clearly $O(N)$ messages are needed in oriented tori as well).

For another important class of networks, namely cliques, orientation does help. While an $\Omega(N \log N)$ lower bound on election in unoriented cliques was shown by Karp et al. [KKM90], an $O(N)$ algorithm for oriented cliques was given by Loui et al. [LMW82].

The question whether orientation helps in hypercubes has remained open. Using an efficient traversal possible in oriented hypercubes, and the construction of Karp et al. [KKM90], election can be performed in $2N \log N$ messages in the oriented hypercube. But, disappointingly, because the hypercube has only $O(N \log N)$ edges, an $O(N \log N)$ complexity is also achieved when the standard algorithm of Gallager et al. [GHS83] is
Figure 1: Message forwarding in the tournament.

the $d$-cube so that the entry node can forward the message in $d$ steps. This announcement would cost $2^d - 1$ messages, leading to an $O(N \log N)$ overall complexity of the election. Similarly, it is too expensive to have to entry node broadcast the tournament message through the $d$-cube; this would also cost $2^d - 1$ messages.

3.2 The Match-Making Technique

The forwarding of the message can be seen as a match-making problem (see Mullender and Vitanyi [MV88]) and can be solved using $O(\sqrt{2^d})$ messages and in $d + 1$ time (which is optimal).

To make a match between the $d$-leader and the entry node, the $d$-leader announces its leadership to all nodes in a $[d/2]$-dimensional face, referred to as the leader's row. The entry node broadcasts the tournament message through a $[d/2]$-dimensional face called its column. As each row intersects each column in exactly one process (as will be shown below), there is one process, called the match process, that receives both the announcement from the $d$-leader and the tournament message. The match process forwards the tournament message further to the $d$-leader via the spanning tree induced by the announcement messages.

**Definition 3.1** Consider the hypercube of dimension $d$.
The row with index $u_{[d/2]} \ldots u_d \ldots u_d$ is the subset of nodes $\{x_0 \ldots x_{[d/2]} \ldots u_{[d/2]} \ldots u_d \ldots u_d\}$.
The column with index $u_0 \ldots u_{[d/2]} - 1$ is the subset of nodes $\{u_0 \ldots u_{[d/2]} - 1 x_{[d/2]} \ldots x_{d - 1}\}$.

**Lemma 3.2** Each node belongs to exactly one row and to exactly one column. Any row intersects any column in exactly one process.

**Proof.** Node $\bar{u} = u_0 \ldots u_{[d/2]} - 1 u_{[d/2]} \ldots u_d - 1$ belongs to the row with index $u_{[d/2]} \ldots u_d - 1$ but not to any other row. This node belongs to the column with index $u_0 \ldots u_{[d/2]} - 1$.
Row $u_{[d/2]} \ldots u_d - 1$ and column $u_0 \ldots u_{[d/2]} - 1$ intersect in process $u_0 \ldots u_{[d/2]} - 1 u_{[d/2]} \ldots u_d - 1$, which is the only process that belongs to both subsets.

Algorithm 2 shows how to broadcast the $\langle \text{ann}, d \rangle$ message through a row using only $2^{[d/2]} - 1$ messages, and without using the canonical node labels. Each process stores the link through which the message was received (variable $\text{fath}_p$), thus building a spanning tree of the row.
\begin{verbatim}
var fath_p: -1..d - 1;

To initiate a broadcast:
begin fath_p := -1; broad([d/2], d) end

Upon receiving (ann,d) via link i:
begin fath_p := i; broad(i, d) end

procedure broad(i):
begin if i > 0
   then send (ann,d) through link i - 1; broad(i - 1, d) end
end

Algorithm 2: Broadcasting in a row.
\end{verbatim}

Lemma 3.3 Alg. 2 broadcasts the message (ann,d) through a row using $2^{[d/2]} - 1$ messages, and builds a spanning tree of the row, rooted at the initiator, of depth $[d/2]$.

Proof. We first show the following by induction on $i$: if $\text{broad}(i, d)$ is executed by process $\bar{u} = u_0..u_{d-1}$, the message $(\text{ann}, d)$ is received exactly once by the processes in $\{x_{0..i-1u_i..u_{d-1}}\} \setminus \{\bar{u}\}$, and a spanning tree on these nodes is built of depth $i$.

Case $i = 0$: Execution of $\text{broad}(0, d)$ is a skip, so no process will receive anything; indeed, $\{u_0..u_{d-1}\} \setminus \{\bar{u}\}$ is the empty set.

Case $i + 1$: Execution of $\text{broad}(i + 1, d)$ by $\bar{u}$ first sends an $(\text{ann}, d)$ message via link $i$, that is, to node $w_i = u_0..u_{i-1}\bar{u}_i u_{i+1}..u_d$. ($\bar{u}_i$ is the complement of $u_i$.) By induction, the subsequent execution of $\text{broad}(i, d)$ in $v$ and $u'$ (the latter upon receipt of the $(\text{ann}, d)$ message from $v$) results in all processes in $\{x_{0..x_{i-1}}u_iu_{i+1}..u_{d-1}\} \setminus \{\bar{u}\}$ and $\{x_{0..x_{i-1}}\bar{u}_i u_{i+1}..u_{d-1}\} \setminus \{w_i\}$

receiving the message once. Consequently, all processes in $\{x_{0..x_{i-1}}x_iu_{i+1}..u_{d-1}\} \setminus \{\bar{u}\}$ receive the message exactly once.

The recursive calls both build a spanning tree of depth $i$, but one is rooted at $u'$ and hence become hooked in at depth $1$, which brings the overall depth at $i + 1$.

Thus, the initialization of a broadcast causes all processes in the initiator's row to receive the message exactly once. The message complexity is $2^{[d/2]} - 1$ because all processes in the row except one receive $(\text{ann}, d)$ once. \qed

A similar procedure is used to broadcast the tournament message through the column of the entry node. This broadcast takes $2^{[d/2]} - 1$ messages.

The tournament between the two $d$-leaders is organized as follows.

1. A $d$-leader $p$ sends a $(\text{tour}, p, d)$ message via link $d$.
2. A $d$-leader announces its leadership in its row by calling $\text{broad}([d/2], d)$.