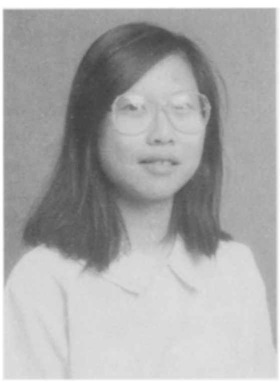
Distributed Computing (1988) 3:19–22

DISTRIBUTED COMPUTING

Distributed election in complete networks

M.Y. Chan[1] **and F.Y.L. Chin**[2]
[1] Computer Science Program, University of Texas at Dallas, Richardson, TX 75083-0688, USA
[2] Computer Studies, University of Hong Kong, Pokfulam Road, Hong Kong



M.Y. Chan received her Ph.D. in 1988 from the University of Hong Kong, and her M.S. and B.A. degrees in computer science from the University of California, San Diego in 1980 and 1981, respectively. She is currently an Assistant Professor at the University of Texas at Dallas.

Francis Y.L. Chin (S'71-M'76-SM''85) received the B.Sc. degree in engineering science from the University of Toronto, Toronto, Canada, in 1972, and the M.S., M.A., and Ph.D. degrees in electrical engineering and computer science from Princeton University, New Jersey, in 1974, 1975, and 1976, respectively. Since 1975, he has taught at the University of Maryland, Baltimore Country, University of California, San Diego, University of Alberta, and Chinese University of Hong Kong. He is currently Head of the Department of Computer Science, University of Hong Kong. He has served as a program co-chairman of the 1988 International Conference on Computer Processing of Chinese and Oriental Languages (Toronto) and the International Computer Science Conference '88 (Hong Kong). His current research interests include algorithm design and analysis, parallel and distributed computing.

Abstract. An improved version of Afek and Gafni's synchronous algorithm for distributed election in complete networks is given and an $O(n)$ expected message complexity is shown.

Offprint requests to: M.Y. Chan

Key words: Distributed election algorithms – Expected message complexity – Synchronous complete networks – Asynchronous complete networks

1 Introduction

This paper presents a modified version of the **synchronous** complete network election algorithm presented by Afek and Gafni (1985). This new algorithm, apart from having a slightly better worst case message complexity, is shown to have an $O(n)$ expected message complexity. The analysis also allows us to demonstrate that an $O(n)$ expected message complexity, alongside an $O(n \log n)$ message and $O(n)$ time complexity in the worst case, is possible for distributed leader election on **asynchronous** complete networks. This complete network result contrasts the $\Omega(n \log n)$ lower bound on average message complexity given for election on asynchronous rings (Pachl et al. 1984).

Afek and Gafni (1985) were the first to address distributed election in the **synchronous** complete network. Previous work on **asynchronous** complete network election includes an algorithm by Korach et al. (1984) which uses $5n \log_2 n + O(n)$ messages and $O(n \log n)$ time. They also showed a lower bound of $\Omega(n \log n)$ on worst case message complexity. Humblet (1984) contributed a time-improved algorithm which uses $2.773n \log_2 n + O(n)$ messages and $O(n)$ time. Afek and Gafni (1985) then gave a slightly improved $2n \log_2 n + O(n)$ message and $O(n)$ time solution. At the same time, they gave their synchronous algorithm which offered $3n \log_2 n$ message and $2 \log_2 n + O(1)$ time complexity, and proved that $\Omega(n \log n)$ messages are necessary, and with such a message complexity, time complexity must be at least $\Omega(\log n)$. All of this work focused on worst case behavior only.

2 The algorithm

We adopt the model of Afek and Gafni (1985). Assume a point-to-point network of n nodes in which every node is connected by $n-1$ bidirectional communication links to all other nodes. There is no shared memory, so nodes communicate only by exchanging messages. Each node has a distinct identification number, and initially has no idea to which node each of its incident links connect. A global clock is connected to all nodes. At the beginning of each clock pulse, nodes receive messages, do local computation, and send messages destined to be received by the next clock pulse. Nodes may start executing the election algorithm either voluntarily at any arbitrary time or upon receiving a message of the algorithm. The same algorithm resides at all nodes.

The algorithm described below is a simple variation on the synchronous complete network election algorithm originally suggested by Afek and Gafni (1985).

Each node maintains three variables: **ID**, **LEVEL** and **STATE**. ID contains the identification number of the node's **owner**. Initially ID is the node's own identity, i.e., initially each node is self-owned. A node can have at most one owner at a time but may change owners in the course of election. The idea is that the node which ends up owning all nodes is the leader. LEVEL is initially 0 and is incremented once every **even** clock pulse. LEVEL basically reflects the total number of nodes which are owned by the node's current owner; in particular, for each node, its owner owns 2^{LEVEL} nodes in all. There are two STATEs a node can be in: **candidate** or **captured**. A candidate node is self-owned and contending for leadership, while a captured node is owned by some other node and is no long in the running for leadership. There are two kinds of messages passed: (LEVEL, ID) messages and KILL messages.

Immediately after LEVEL increase, at the beginning of every even clock pulse, a candidate node tries to, in fact, double the extent of its ownership

Algorithm

```
ID:=identity of the node
LEVEL:=0
STATE:=candidate
OWNER-LINK:=nil
for each clock pulse do
    if clock pulse is even then
        receive all KILL messages
        if KILL is received then STATE:=captured
        if 2^LEVEL ≥ n then
            if STATE=candidate then YOU ARE THE LEADER, STOP
            else LEADER IS ACROSS OWNER-LINK, STOP
        LEVEL:=LEVEL+1
        if STATE=candidate then
            /*you now own 2^(LEVEL-1) nodes, try for 2^(LEVEL-1) more*/
            send (LEVEL, ID) across up to 2^(LEVEL-1) unmarked links
            mark these links
    else /*clock pulse is odd*/
        receive all (LEVEL, ID) messages
        let (LEVEL*, ID*) be the lexicographically highest (LEVEL, ID) just received
        let L* be the link across which (LEVEL*, ID*) was received
        for each (LEVEL, ID)≠(LEVEL*, ID*) just received do
            send KILL to its initiator
        if (LEVEL*, ID*)<(LEVEL, ID) then
            send KILL across L*
        else
            if OWNER-LINK≠nil then send KILL across OWNER-LINK
            STATE:=captured
            OWNER-LINK:=L*
            (LEVEL, ID):=(LEVEL*, ID*)
od
```

Fig. 1

by sending its (LEVEL, ID) pair across $2^{\text{LEVEL}-1}$ unmarked links which then become marked for this node. A candidate is allowed to continue as candidate at the next even pulse only if no KILL message is received then. Otherwise, the candidate node becomes captured. If successful (i.e., no KILL received), the candidate will end up being the sole owner of 2^{LEVEL} nodes.

At the beginning of every odd round, each node considers the **lexicographically** highest (LEVEL, ID) pair then received. In response to all other (LEVEL, ID)s received, a KILL message is returned. If the highest (LEVEL, ID) is lower than the current (LEVEL, ID) of the node, a KILL is also issued to it. Otherwise, the initiator of the highest (LEVEL, ID) message becomes the owner of the node, the node's (LEVEL, ID) is changed to this higher (LEVEL, ID), the node becomes or stays captured, and the KILL is issued instead to the node's previous owner. In other words, each captured node tries to establish for itself at most one owner.

Each node stops the election process when it discovers its owner owning all of the nodes in the network. Details of the algorithm are found in Fig. 1:

Remark 1. If, instead of using explicit KILL messages and implicit ACK (acknowledgement of owner) messages, we were to use explicit ACK messages and implicit KILL messages, i.e. each node would send an ACK message to acknowledge its current owner at every odd clock pulse and each candidate would only remain candidate upon receiving ACK messages from all the nodes it has sent its identity to, then we would obtain an algorithm that is still slightly **different** from Afek and Gafni's algorithm. The difference is that each candidate node is the confirmed owner of all previous nodes it has captured and not just those which it has recently captured. Another way of looking at it is that owners in our algorithm know when they have been displaced, but in Afek and Gafni's algorithm, displaced owners are not informed. Because of this difference, we can obtain an explicit-ACK-implicit-KILL algorithm (the number of (LEVEL, ID) messages sent by candidate nodes is also changed) which is better in both message and time complexity than Afek and Gafni's algorithm. However, the improvements are by constant factors and this algorithm does not have an $O(n)$ expected message complexity. See Chan and Chin (1986) for the details of the $1.89n\log_2 n+O(n)$ message and $1.26\log_2 n+O(1)$ time solution. The presented algorithm represents a constant factor improvement in worst case message complexity over Afek and Gafni's synchronous algorithm.

Remark 2. The expected case analysis of the presented algorithm can be carried over to the analysis of Afek and Gafni's algorithm, and Afek and Gafni's algorithm can be converted into an asynchronous algorithm using $4n\log_2 n+O(n)$ messages and $O(n)$ time while maintaining the $O(n)$ expected message complexity. See Chan and Chin (1986) for the details. The conversion involves having nodes consult their current owners whenever they receive a (LEVEL, ID) message to decide whether they should keep or change their current owner. This is necessary because, in the asynchronous model, there is no global clock and candidate nodes do not enlarge their ownerships at the same pace. A side-effect of this consultation with the owner is that owners will know when they are being displaced, so this asynchronous algorithm can be viewed as the asynchronous counterpart of our algorithm as well and explains the message complexity claimed.

Theorem. *The algorithm elects a leader using $2n\log_2 n$ messages in the worst case, $O(n)$ messages on the average, and $2\log_2 n+O(1)$ time.*

Proof

Correctness. First observe that there can be at most one leader, since ownership of a node is disjoint and only the node which owns all others can be leader. Secondly, it is not possible for all nodes to be captured, i.e. for there to be no leader. To see this, suppose all nodes are captured and consider the node with the lexicographically highest (LEVEL, ID) at the time of capture. This node must have been captured because of a candidate node with higher (LEVEL, ID) which at its time of capture may have an even higher (LEVEL, ID); this gives an obvious contradiction. Furthermore, each candidate will either be captured or increase the extent of its ownership within two clock pulses. Thus, exactly one node will be elected as leader.

Worst case message complexity. There are at most $n/2^i$ candidates which disjointly own 2^i nodes. Each of these candidates, for $i=0,1,\ldots,\lceil\log_2 n\rceil-1$, will send out at most 2^i (LEVEL, ID) messages in an attempt to double the extent of its ownership. There can be at most one KILL message associated with each (LEVEL, ID) sent. Thus, the message complexity is

$$2\sum_{i=0}^{\lceil \log_2 n\rceil -1} 2^i \cdot \frac{n}{2^i} \leq 2n \log_2 n.$$

Time complexity. The eventual winner of the election must be among the first nodes to awake. With two clock pulses per level increase and $\lceil \log_2 n\rceil$ levels, the time complexity is $2\log_2 n + O(1)$.

Average case message complexity. The crux of the proof lies in determining the upper bound on the expected number of candidates to survive each level. To this end we have the following lemma.

Lemma. *Let a be the number of nodes disjointly owned by each candidate at the start of level i and b be the number of messages sent by each candidate at the start of level i. Then, the expected number of candidates to start level $i+1$ will be at most $1+\frac{n}{ab}$.*

Proof. Let $x_1, x_2, \ldots, x_m$ be the candidates to start level i arranged in descending ID order where $ma \leq n$, and p_j be the probability that x_j survives level i. If x_j is to survive level i, it must send messages to b nodes other than those already owned by $x_1, \ldots, x_{j-1}$. Hence,

$$p_j \leq \binom{n-ja}{b} \Big/ \binom{n-a}{b}$$

Since

$$\binom{n-ja}{b} \leq \binom{n-ja+i}{b} \quad \text{for } i \geq 0,$$

$$p_j \leq \sum_{i=1}^{a} \binom{n-(j-1)a-i}{b} \Big/ a\binom{n-a}{b}$$

So the expected number of candidates to start level $i+1$ is

$$\begin{aligned}
\sum_{j=1}^{m} p_j &= 1+\sum_{j=2}^{m} p_j \leq \\
&\quad 1+\sum_{j=2}^{m}\sum_{i=1}^{a} \binom{n-(j-1)a-i}{b} \Big/ a\binom{n-a}{b} \\
&\leq 1+\sum_{k=a+1}^{ma} \binom{n-k}{b} \Big/ a\binom{n-a}{b} \\
&\leq 1+\sum_{k=a+1}^{n} \binom{n-k}{b} \Big/ a\binom{n-a}{b} \\
&\leq 1+\binom{n-a}{b+1} \Big/ a\binom{n-a}{b} \\
&= 1+\frac{n-a-b}{a(b+1)} \leq 1+\frac{n}{ab} \qquad \square
\end{aligned}$$

Thus, the expected number of candidates to begin level 1 is at most n, level 2 is at most $n/2$, and with $a=b=2^{i-1}$ using Lemma, level $i+1$ is at most $1+n/2^{2i-2}$ for $i=2, \ldots, \lceil \log_2 n\rceil -1$. Recall that the number of (LEVEL, ID) messages sent by each candidate upon level increase is $2^{\text{LEVEL}-1}$ (so at level $i+1$, each candidate sends 2^i (LEVEL, ID) messages), and for each (LEVEL, ID), there is at most one KILL message. Hence, the expected number of messages is at most

$$4n + \sum_{i=2}^{\lceil \log_2 n\rceil -1} 2^{i+1}\left(1+\frac{n}{2^{2i-2}}\right) = O(n). \qquad \square$$

References

Afek Y, Gafni E (1985) Time and message bounds for election in synchronous and asynchronous complete networks. Proc ACM Symp Principles Distributed Comput, Minacki, Ontario (August 1985) pp 186–195

Chan M-Y, Chin F (1986) Expected and worst performance for election in synchronous and asynchronous complete networks. Tech Rep TR-20-86, Centre of Computer Studies and Applications, University of Hong Kong (October 1986).

Humblet PA (1984) Selecting a leader in a clique in $O(N \log N)$ messages. Proc 23rd Conf on Decision and Control, Las Vegas, Nevada (December 1984) pp 1139–1140

Korach E, Moran S, Zaks S (1984) Tight lower and upper bounds for some distributed algorithms for a complete network of processors. Proc ACM Symp Principles Distributed Comput, Vancouver, BC (August 1984) pp 199–207

Pachl J, Korach E, Rotem D (1984) Lower bounds for distributed maximum-finding algorithms. J ACM 31:905–918