Leader election in the set of permutation graphs with $O(\mid E \mid +n)$ messaging complexity

Sai Ganesh
email : ssai@wipsys.soft.net *

September 20 1998

Abstract

Leader election is a fundamental problem in distributed systems. The elected leader of a network may be responsible for coordinating various events in the system or solving a critical problem.

This paper presents a simple, efficient algorithm to elect a leader in the set of permutation graphs which has n processors in an undirected, asynchronous, connected network $G = (V, E)$. Reliable communication among nodes in which each node supporting FIFO queuing mechanism is assumed for the problem.

A permutation sequence π basically defines the edge-connectivity of the network in that an edge appears in the graphs $G = (V, E)$ only if two numbers i and j appear interchanged in the permutation sequence π. Making use of some of the properties of permutation graphs as defined below, this wave algorithm provides a messaging complexity of $O(\mid E \mid +n)$ with a worst case time complexity $O(n^2)$.

By proposing the algorithm for a single permutation, we cover a large and wide variety of network connectedness for all permutations possible.

1 Introduction

Leader election is a fundamental problem in distributed systems. The elected leader of a network may be responsible for coordinating various events in the system or solving a critical problem. The problem involves in that given a set of

*I like to express my thanks to Meena and KamaKoti both of MatScience, Chennai for giving me the excellent oppurtunity to work at MatScience and for greatly assisting this work

prospective candidates, the technique is to be call the node n_i with the extreme identification id_i to be the leader of the network.

Consider a undirected, connected network $G = (V, E)$ which has a total of number of processes $n =\mid E \mid$ in the system. Consider a permutation graph defined from a permutation sequence π as defined below :

A permutation sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ basically defines the edge-connectivity of the network in that an edge appears in the graph $G = (V, E)$ only if two numbers i and j appear interchanged in the permutation sequence π. More formally, the graph $G[\pi] = (V, E)$ is defined as follows :

$$V = \{1, 2, \ldots, n\} \text{ and}$$
$$ijE \Leftrightarrow (i - j)(\pi_i^{-1} - \pi_j^{-1}) < 0$$

where $\pi_i^{-1}(k)$ denotes the position of number k in the permutation sequence π. The undirected graph $G \cong G[\pi]$ is called a permutation graph for the given permutation sequence π[Golumbic].

Making use of some of the assumptions and properties of permutation graphs as defined below, this wave algorithm provides a messaging complexity of $O(\mid E \mid +n)$ with a worst case time complexity $O(n^2)$.

Some of the properties/characteristics of permutation graphs used in the conception of idea of the algorithm implementation are presented below.

1. Permutation Graphs are transitively oriented in that $ij, jk \in$ E $\Rightarrow$ik $\in$ E. Also this implies that $\pi_i^{-1} > \pi_j^{-1} > \pi_k^{-1}$.

2. Let $\pi = [\pi_1, \pi_2, \ldots, \pi_n]$ be a permutation sequence of $1, 2, \ldots, n$. Let p_i denote the number of integers less than and to the right of i and let q_i denote the number of integers greater than and to the left of i in sequence π. Then the sequence is such that : $\pi_i^{-1} + p_i = i + q_i$ (or $\pi_i^{-1} - i = q_i - p_i$).

3. For the numbers $1, 2, \ldots, n$, the following permutations offer :

1. hspace*6 Complete graph $[n, n-1, \ldots, 1]$
2. hspace*6Graph with isolated vertices $[1, 2, \ldots, n]$
3. hspace*6Two disjoint complete graphs on r and $n - r$ hspace*6$[r, r - 1, \ldots, 1, n - 1, \ldots, n - r]$ for $r > n - r$
4. hspace*6Two disjoint sets in which each of them is connected graph hspace*6with one set r and the other with $n - r$ vertices hspace*6$[r, r - 1, \ldots, 1, n, n - 1, \ldots, n - r]$ for $r > n - r$
5. hspace*6A star connectivity : $[n, 1, 2, \ldots, n - 1]$

An algorithm that elects a leader in general network with $O(\mid E \mid +n)$ messaging complexity is proposed by Gallager[Gallager]. It is shown that for a general graph, atleast $\Omega(\mid E \mid +n \log n)$ messages must be exchanged which is the lower bound to the problem of election in the worst case[KMZ]. Other algorithms with lesser messaging complexity for a specialized and restricted subsets of network topology are proposed[KMZ, DKR, Peterson].

2 Model and Assumptions

This section defines the model for the proposed computing along with the set of assumptions made of the system and algorithmic assumptions.

Consier a graph $G = (V, E)$ which is a finite, undirected, connected network. Consider a permutation graph which is formed from permutation sequence. Following assumptions are made for working in this model.

1. Communication among the processes is asynchronous with no assumed timeout for messages to arrive at the destination node. However, total reliability with no loss of messages is assumed

2. Each node holds as the initial input the permutation sequence π and hence understands and identifies the network connectivity

3. Each node (we will be using the term node and process interchangeably) supports a FIFO order of queueing mechanism for the reception of messages

4. Each node n_i is idenfied by a two-tuple ¡nn_i, id_i¿ where nn_i denotes the node number of that node and id_i denotes the unique identifier Node number nn_i is used in identifing a particular node in the permutation sequence π. The node number maps one-to-one to the identifier

5. Each node n_i identifies its neighbouring nodes/ports using their node number nn_k ($k \in$Neig$_i$)

6. Each node n_i communicates only with the nodes of depth 1 from itself i.e. only with its neighbouring nodes. However, decision at node n_i can take place based on this sequence

7. Each node starts off spontaneously or on reception of any message. If started on reception of a message, a node should execute the local initialization procedure defined in 1 in the algorithm before further processing. As an alternative, all nodes could be triggered at once by an initial dummy init message which would cost an additional $2 \mid E \mid$ messages

8. There is no limitation on the number of messages that can be sent and on the size of each of these messages

3 Algorithm

This sections proposes the algorithm for the leader election for the above defined permutation graphs.

3.1 General Outline

Each node is initially holds the permutation sequence π such that they understand and identify the network connectivity among the various nodes. Nodes n_i initially broadcast their identities id_i to all it neighbours $Neig_i$. Each node keeps a count of the number of initial identifier messages received.

Each node n_i elects a node called $dnode_i$, its decision node to communicate all success reports to it. The decision node $dnode_i$ for a node n_i is the one which has the largest degree among its neighbours. The largest degree node as decision node is due to the obvious reason that its capable of deciding better than itself and any other neighbouring nodes. The crux of the problem lies in selecting the decision node among all its neighbours.

Once the node receives identifiers from all its neighbouring nodes, it decides as to which is the largest identifier among them. It then reports this information to its decision node. The decision node expects as many as the number of identifier received messages before it passes its decision to its decision node and this goes on till the root node. The root node, however, waits till all the $n-1$ reports are received and makes a decision as to which can be the right leader for the network.

3.2 Formal Description

Algorithm defined below is defined in five steps. The first step defines the initialization procedure the node needs to execute. Second, third and fourth steps define the action when a node receives identifier information, report and final broadcast messages respectively.

Step 1 (initilization) : Initializes maximum identifier max_id_i and sends it own identifier id_i to all its neighbours.

Step 2 (**idinfo** message received) : Increments $received_i$, checks for the maximum of the identifier received and if necessary reports it to its decision node.

Step 3 (**report** message received) : Checks for the maximum identifier and reports further to its decesion node. If this is the node of highest node number nn_i, then it terminates and reports to all the other nodes in the network.

Step 4 (**found** message received) : Forwards this found message to all its neighbours.

Step 5 (**find_dnode** procedure) : This procedure elects the best possible decision node for this node in the network. This uses the permutation sequence to check for the largest connectivity node which is adjacent to this node.

Algorithm Elect_Leader
Inputs to each node i : Permutation sequence $\pi = [\pi_1, \pi_2, \ldots, \pi_n]$
Output : At termination, each node n_i has tuple ¡nn_l, id_l¿ of the leader

1. // Initilizaton at each node n_i
   begin
   $max\_id_i := id_i$;
   max_tuple_i := ¡i, id_i¿;
   call local procedure find_dnode();
   send idinfo i, id_i ton_k ∀k∈Neig$_i$;
   end

2. // Upon reception of **idinfo**k, id_k
   begin
   received$_i := received_i + 1$;
   if $(id_k > max\_id_i)$ **then**
   begin
   $max\_id_i := id_k$;
   max_tuple_i := k, id_k ;

if $(received_i = \#Neig_i \wedge dnode_i \neq i)$ **then**
send report (max_tuple_i) to $dnode_i$;
end

3. // Upon reception of **report**k, id_k
   begin
   if(id$_k > max_id_i$) **then**
   begin
   $max_i := id_k$;
   max_tuple_i := k, id_k ;
   report_msg$_i := report_msg_i + 1$;

if $(i \neq n \wedge report\_msg_i \leq n-1 \wedge dnode_i \neq i)$ **then**
send report $((max_tuple_i)to$node$_i$;
else if $(i = n \wedge report\_msg_i = n-1)$ **then**
send found $((max\_tuple_i)$ to $\forall k \in$Neig$_i$;
end
end

4. // Upon reception of **found**k, id_k
begin
sendfound$k$,id$_k$ $to\forall k \in$Neig$_i$;
end

5. // Procedure to elect decision node $dnode_i$
procedure find_dnode()
begin
$p_i := Number\,of\,elements\,to\,the\,right\,and\,less\,than\,i\,in\,\pi$;
$dnode_i :=$ MAX(j) ¡ $i\ \forall$j$torightof$i$in\pi\vee$i;

for j := 1 to π_i^{-1} in step 1 **do**
if $(\pi_j > dnode_i \wedge \pi_j \not< i)$ **then**
$dnode_i := \pi_j$;
end

Theorem 3.1

Algorithm Elect_Leader elects a leader in permutation network with a messaging complexity $O(\mid E \mid +n)$ with worst case time complexity $O(n^2)$.
Proof.

For any connected network, node

n_i broadcasts its identity id_i to all its neighbour and all of the neighbours of n_i send their identities to n_i. This makes up for the initial $\mid E \mid$ messages. Also, the final **found** messages add to $\mid E \mid$ number.

Corollary 3.2

Each node n_i recieves as many as $Neig_i$ **report** messages before it decides.

Corollary 3.3

The number of **report** messages exchanged in the whole of network is n and the node with largest node number nn_i waits for all the $n-1$ reports to arrive before it decides the leader. Hence the total messaging complexity is $O(\mid E \mid +n)$.

Lemma 3.4

Algorithm Elect_Leader is a wave algorithm.
Proof.

Wave algorithm are those which satifies the following three requirements

[Tel] :

Lemma 3.5

Termination Computation at each node is defined and the algorithm takes a finite number of steps before termination.
Proof.

Given the permutation sequence π, each

node n_i calculates its decision node $dnode_i$ first and sends its identity id_i to all its neighbours $Neig_i$. Assumuing asynchronous, FIFO, no loss of information, each node exits on the reception of id_i messages after $2n-1$ pulses from the start. A node triggers reporting of max_id_i by sending **report** messags to its $dnode_i$ and thus avoids putting other nodes in a state of indefinite wait. Since 'decision nodes' expects as many **report** messags as **idinfo** messages sent out, they are definite to terminate. Nodes that don't take part in decision accept **found** messages at the end and hence terminate.

Potential deadlock could occur only when one node indefinetly waits for the other nodes to report and the other nodes wait for the former one for the same reason. However, as from the step 2 in the algorithm, this cannot happen since once all **id**s are received, this node decides and reports which results in switching to the next state.

Claim.

Each computation at each node contains a decide event. This is the **report** event which casually preceeds events triggered from the other nodes.

Claim.

Each decide event which is **report** messaging, casually is preceeded by a trigger of **idinfo** message $\forall$i $\in Neig_i$. This proves the wave algorithm characteristics for our algorithm.

Lemma 3.6

Each node n_i elects a single node called the decision node to announce its report. This election is done at $2n-2$ pulses from start.

Proof.

The first two steps in the procedure **find_dnode** can be executed at

a worst case of $n-1$ pulses from the start. The third step (the for loop) runs through $n-1$ steps.

Corollary 3.7

Each node n_i receives as many as $\#Neig_i$ **report** messages before it decides.

4 Leader election in specific networks

This section identifies a couple of examples on which this algorithm runs in the permutation network. The examples considered are the ring and the complete clique topology since they seems to fit in the extreme case for the messaging complexity.

As pointed out in section 3, these networks can also form permutation networks - one among the very large subsets of permutation network connectivity possible.

4.1 Ring network

Analysis of the algorithm shows that in ring networks, a maximum of $2 \mid E \mid$ messages are initially exchanged for each of the nodes to identify the **id**s of their neighbouring nodes. Remember, even this can be avoided if initially its assumed that each node is aware of all the **id**s of its neighbouring nodes.

Each node then reports its opinion of maximum identifier to its decision node which happens to be its adjacent. The trick here is that the decision node always has a greater node number nn_i than itself so the whole system actually behave like a token-ring system passing its token to its neighbouring node. This procedure however stops at the greatest node since this decides, finally. The message complexity, as we can see, is $O(n)$.

4.2 Complete network

In complete network, the leader is elected in the second step itself since all the nodes receive **id**s from all other nodes in the network. Hence, the decision

node is the same for all the other nodes in the network (except for the decision node itself) in which case, a **found** message is sent announcing to the whole network of the leader election. The message complexity in this case is $O(\mid E \mid +n), which is the best possible.$

References

[M.C.Golumbic] Algorithmic Graph Theory and Perfect Graphs Academic Press Inc.

[R.Gallager, R.G., Humlet, P.A. and Spira, P.M.] A Distributed Algorithm for Minimum Weight Spanning Trees, ACM Trans. Program. Syst., Vol. 5. pp 66-77 Jan,1983

[E.Korach, S.Moran, S.Zaks] Tight Lower and Upper Bounds for some Distributed Algorithms for a Complete Network of Processors, Proc. of the Third Annual ACM symp. on Principles of Distributed Computing (1984) pp. 199-207

[D.Dolev, M.Klawe and M.Rodeh] An $O(n \log n)$ Unidirectional Distributed Algorithm for Extrema-Finding in a Circle, Journal of Algorithms, Vol. 3, Sep,1982

[G.L.Peterson] Efficient Algorithms for Elections in Meshes and Complete Networks, University of Rochester, Aug,1984

[Gerard Tel] Introduction to Distributed Algorithms Cambridge University Press.

[Yossi Matias and Yehuda Afek] Simple and Efficient Election Algorithms for Anonymous networks, Lecture Notes on Computer Science, Distributed Algorithms [1997]

[Hagit Attiya] Constructing Efficient Election Algorithms from Efficient Traversal Algorithms, Lecture Notes on Computer Science, Distributed Algorithms [1997]

[Gurdip Singh] Efficient leader election using sense of direction, Distributed Computing 10 [1997]