An Efficient Cellular Automata based Leader Election Scheme

Monalisa Dey
Computer science & Engineering Dept
JIS College of Engineering
Kalyani, Nadia, WB-741235, India

Prasenjit Dey
Nivio Technology Pvt. Ltd.
Gurgaon, Haryana, 122016.

ABSTRACT

This work reports an efficient scheme of electing a leader, in a fully connected distributed system, having n number of nodes. In the proposed scheme, the system state is modeled using Cellular Automata. Each node is initialized with status information. This information has to be maintained by the nodes at all times so that they are aware of the current working coordinator in the system. The proposed scheme requires only O(n) messages for decision making.

Keywords

Distributed systems, cellular automata**.**

1. INTRODUCTION

In a distributed system, where nodes (processors) often compete as well as cooperate to achieve a common goal, it is sometimes needed to reach an agreement among the nodes. Such a common problem is the coordinator/leader election problem. It is defined as the process of choosing a node from among a group of nodes in a distributed system to act as the central coordinator. A leader or coordinator is elected to coordinate a certain task. Such tasks may include mutual exclusion handling or failure recovery.

The participating nodes can compete or cooperate among themselves for such decision making. This results in voluminous information exchanges. A number of such algorithms have been proposed in literature [1], [2], [3]. However the worst case message exchange complexity of these algorithms are $O(n^2)$.

In this paper, a different approach is presented to address this problem. The system state is modeled by initializing with the status information at each node and each node maintains this model so that they are aware of the current coordinator.

Using this approach it is possible to elect a leader much efficiently with minimum number of message exchanges.

In the following sections, first the proposed approach is described, and then it is shown how the system model is developed using CA (Cellular Automata).

2. PROPOSED APROACH

In the proposed model, each node records an n-bit status information for n number of nodes in the system. This information represents the overall system state. If a node, say p_i is the current coordinator, then the i^{th} bit from the right in the status information is set to '1' and rest of the bits are set to '0'. For example, if the status information recorded in any node of a distributed system having 5 nodes is '0 0 1 0 0', it implies that node p_3 is the current coordinator. The system state needs to be kept consistent at all times. The whole process of electing a coordinator is described next:

- First the system state is modeled by initializing each node with the status information and each node maintains this model so that they are aware of the current working coordinator.
- When a node p_i detects that the current coordinator say p_f is not responding, p_i then utilizes the proposed scheme for coordinator selection to elect a new coordinator (p_n). Practically, the proposed approach does not involve any communication among the nodes for computing the coordinator node id. The node p_i can unanimously compute the coordinator node id p_n.
- Once node p_i computes the new coordinator node id (p_n), it sends a request message to node p_n to ask whether it is ready to be the new coordinator. The message delivery time is bounded and is taken as T. If p_n does not reply within T time units then p_i decides that p_n is non-existent, it updates the status information, and starts the process from the beginning considering p_n as the current failed coordinator. The failed coordinator p_f is also detected and the status information is modified to reflect that information.
- If p_n exists but it decides not to act as the coordinator then it replies back to p_i that it is not ready. In this situation, p_i continues to find another coordinator considering p_n as the current failed coordinator.
- If p_n replies positively to p_i then p_i updates its system state and considers p_n as the new coordinator, broadcasts this information throughout the network so that all other nodes also update their states accordingly.
- If p_f suddenly wakes up it broadcasts a message throughout the network that it is awake, so that all the other nodes can update their state accordingly. p_n sends a message to p_f declaring itself as the currently working coordinator. During this time if any node sends a request to p_f to be the new coordinator it disagrees because it is still not ready with a valid system state.
- I n the initial state, when no coordinator has been elected, a node, after entering a system, broadcasts a message throughout the network

that it is alive. I t waits for 2T time units for reply from the current coordinator. If it does not receive a reply it assumes that there is no coordinator. The new node then broadcasts a message to inform the other nodes in the system about this situation. If this is really the initial state, they do not respond. The new node, after 2T time units, broadcasts a message declaring itself as the new coordinator and wais for 2T time units.

- If this is not really an initial state, all the nodes send the coordinator information to the newly added node.

Example 1: The following example illustrates the scheme precisely:

Let us consider a distributed network having four nodes. Node p_1 is assumed to be the current coordinator. Thus the current state of each node is '0 0 0 1'. The i^{th} bit (from the right) corresponds to i^{th} node. Let us consider node p_1 has failed to act and node p_3 has detected it because it did not get any response when it had made request to the coordinator for some services. Node p_3 then follows the proposed scheme and elects node p_2 as the new coordinator. Node p_3 then informs this to node p_2. Node p_2 agrees to act as coordinator and responds affirmatively to node p_3. In the next step node p_3 broadcasts the information of new coordinator so that all nodes can modify their system state accordingly. The whole process is described in Fig. 1.

3. SYSTEM MODEL

A different approach is presented here to address the problem of leader election. A model of the system is developed which is the partial view of the global state of the system. Each node is supposed to maintain the model and update it accordingly.

The system under consideration is a distributed network consisting of n different nodes that are interconnected through network. The number of nodes may vary from a few to a few thousands. The proposed solution scheme is designed around the Cellular Automata (CA), an unconventional modeling tool invented by von Neumann.

4. INTRODUCTION TO CA

In its simplest form cellular automata evolves in discrete space and time, and can be viewed as an autonomous finite state machine (FSM). Each cell stores a discrete variable at time t that refers to the present state (PS) of the cell. The next state (NS) of the cell at (t + 1) is affected by its state and the states of its neighbors at time t. We will use a two state three neighborhood CA which can have two states (0 and 1) and the the next state function and St i-1, St i and St i+1 are the present states of left, self and right neighbors of cell i. The collection of the states of all cells is called a state of the n-cell CA at time t.

The next state function of the ith CA cell can be expressed in the form of a truth table shown. The decimal equivalent of the 8 outputs is called Rule Ri. In a 2-state 3-neighborhood CA, there can be 28(256) rules. Two such rules 30 and 18 are illustrated in Fig. 2. The first row lists the possible 23(8) combinations of present states of (i − 1)th, ith and (i + 1)th cells at time t. The last two rows indicate the next states of the ith cell at time (t + 1).

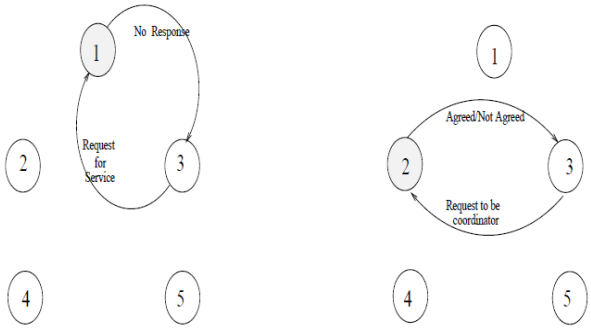

Fig. 1. The Process Model

5. CA BASED MODEL

In the current application, for a system of n nodes, an n-cell null boundary CA, configured with rule 18 at each node. The current status information maintained at a node is considered as the present state (PS) of the CA at that node. During the coordinator election process, the node that identifies the absence of a coordinator runs the CA for a single step considering the current system status information as the initial seed and the next state (NS) of the CA correctly indicates the newly elected coordinator id.

Example 1 Revisited:

Consider the distributed system as described in example 1 earlier. After node p3 detects that p1 is not responding it runs the 4-cell CA (configured with rule 18) for one step and reaches the pattern '0 0 1 0'. Here we will take node p2 to be the next coordinator if it agrees. If not we will carry on the selection process considering node p2 as the current failed coordinator and the failure information of node p3 is also reflected in the state information. If in the beginning node p3 was the non- responding coordinator and node p1 ran the CA then in the next step the pattern we get is '1 0 1 0'. In this case we will always select the node with the higher Id to be the next coordinator step.

6. EXPERIMENTAL RESULTS

Previously, Kalyan Mahato and Sukanto Das [4], had proposed a cellular automata based approach to elect a coordinator, however they had used rule 30. The CA configured with rule 30 was ran for n/2 steps (n is the total number of nodes in the system). From the pattern they got They identified the middle cell of the sequence which has largest number of active neighbouring cells and then considered the node of the corresponding mid cell as the new

PS	111	110	101	100	011	010	001	000
RMT	(7)	(6)	(5)	(4)	(3)	(2)	(1)	(0)
NS	0	0	0	1	1	1	1	1
NS	0	0	0	1	0	0	1	0

Fig. 2. The RMT's for Rule 30 and 18

new coordinator.

The two approaches are compared in Table I in terms of the maximum number of active cells present in the next state pattern generated after running the CA rule 30 for n/2 steps and CA rule 18 for 1 step.

The 1st column represents the length of the CA that corresponds to the number of nodes in the system, the 2nd column gives the present state of the system, the 3rd column represents the next state pattern generated. Columns (3-5) give the results of the proposed scheme using rule 18. Column 6 shows the next state pattern generated after running the CA for n/2 steps applying rule 30, column 7 and column 8 shows the total number of conflicts and maximum number of active cells, respectively for scheme [4].It is clearly seen that the number of conflicts that are arising in the next state pattern generated after running the CA for 1 step using rule 18 are much less than the number of conflicts that are arising after using rule 30. Also the maximum number of active cells in the next state pattern generated is much more in number when rule 30 is used. For example, in a CA with length, say 32, the maximum number of active cells in the next state pattern when rule 18 is used is 2. Whereas for rule 30 it is 15. As the length of the CA goes on increasing the number of active cells increases for rule 30 and thus the problem of selecting the same coordinator increases further. Whereas for rule 18 the maximum number of active cells are always 2. Thus the proposed scheme shows more clarity and efficiency compared to the scheme [4].

7. PERFORMANCE STUDY

- The approach proposed in [4] had a certain random nature associated with it. After running n/2 steps the numbers of active neighbouring cells (i.e. number of 1's) are much more compared to the proposed approach. Thus it may be confusing so as to which active cell should be selected.
- In [4] using rule 30, from the pattern obtained, the middle cell of the sequence which has largest number of active neighbouring cells was identified and then considered the node of the corresponding mid cell as new coordinator. This means they had to find out the largest common subsequence of 1's and then identify the middle cell. This increases the complexity of searching in their scheme. Whereas in the proposed scheme the method of selecting a new coordinator is much more simple. If more than one active cell is coming in the next state pattern, we just select the node having the higher node id as the new coordinator.
- The scheme proposed in [4] doesn't guarantee that the currently failed coordinator won't be elected immediately since after running the CA for n/2 steps the cell which was active in the present state may also be active in the next state. For example, the next state corresponding to the present state '1 0 0 0 0' (after the CA runs n/2 steps), is '1 0 1 0 0'. Here, the fifth cell is still active and it might have a chance to be selected as the coordinator again. But the proposed approach ensures that a failed coordinator won't be elected again. As seen from the column 2 and column 3 of Table 2, the bit which was active before will not be active in the next state pattern generated. It is also to be noted that only a single bit is active in most of the cases and in some cases there are more than one active bit. However, the numbers of active bits are at most two in all cases.
- In scheme [4], the CA has been run for *n*/2 steps, so the run time and hence power consumption is more as compared to our approach where the CA is run for just a single step.

Message Complexity:

In the proposed model, let the elected coordinator is not alive and overall, say d, number of nodes are dead where ($1 \leq d \leq n-2$) then node p_i requires (d+2) messages to select new coordinator and (n-d-1) for broadcasting the information. Therefore, the message requirement is n+1, i.e. message complexity is O(n).

8. CONCLUSION

In a distributed system, a very common problem is process of choosing a node from among a group of nodes in a distributed system to act as the central coordinator. A leader or coordinator is elected to coordinate a certain task. So, an efficient leader election algorithm is essential.

In the proposed work, a cellular automata based approach to elect a leader with minimum amount of confusion is presented. A null boundary CA configured with rule 18 is used for this purpose. Previously in the approach [4] rule 30 was used, however as shown in the previous sections, the approach where rule 18 is used is more efficient and has more clarity than the previous approach. Results show that as the length and hence the number of nodes goes on increasing the number of active cells increases for rule 30 and thus the coordinator selection procedure among those active cells become more intensive. Whereas for rule 18, the maximum number of active cells are always 2. Thus the proposed scheme has much more clarity and is less complex than the scheme in [4]

.

TABLE I. Comparison of the Proposed Approach with the Previous Approach

Length	Present State	Rule 18			Rule 30		
		Next State	*#conflicts*	*Max 1's*	*Next State*	*#conflicts*	*Max 1's*
4	0001	0010	0	2	0110	1	2
	0010	0101	1		1100	1	
	0100	1010	1		1001	1	
	1000	0100	0		1010	1	
8	00000001	00000010	0	2	00011001	2	4
	00000010	00000101	1		00110011	3	
	00000100	00001010	1		01100100	2	
	00001000	00010100	1		11001000	2	
	00010000	00101000	1		10010001	2	
	00100000	01010000	1		10100010	2	
	01000000	10100000	1		10000100	1	
	10000000	01000000	0		10101000	2	
16	0000000000000001	0000000000000010	0	2	0000000110010001	3	7
	0000000000000010	0000000000000101	1		0000001100100000	2	
	0000000000000100	0000000000001010	1		0000011001000010	3	
	0000000000001000	0000000000010100	1		0000110010000100	3	
	0000000000010000	0000000000101000	1		0001100100011100	5	
	0000000000100000	0000000001010000	1		0011001000111001	6	
	0000000001000000	0000000010100000	1		0110010001110000	5	
	0000000010000000	0000000101000000	1		1100100011100000	5	
	0000000100000000	0000001010000000	1		1001000111000001	5	
	0000001000000000	0000010100000000	1		1010001110000010	5	
	0000010000000000	0000101000000000	1		1000011100000100	4	
	0000100000000000	0001010000000000	1		1010111000001000	5	
	0001000000000000	0010100000000000	1		1010110000010000	4	
	0010000000000000	0101000000000000	1		1010000000100000	2	
	0100000000000000	1010000000000000	1		1011110001000000	5	
	1000000000000000	0100000000000000	0		1010101010000000	4	

Length	Present State	Rule 18			Rule 30		
		Next State	#conflicts	Max 1's	Next State	#conflicts	Max 1's
32	00000000000000001	00000000000000010	0	2	10010001111001100	7	15
	00000000000000010	00000000000000101	1		00100001011111000	7	
	00000000000000100	00000000000001010	1		01000010111100011	7	
	00000000000001000	00000000000010100	1		10000101111001000	7	
	00000000000010000	00000000000101000	1		00001110011110101	11	
	00000000000100000	00000000001010000	1		00111001111100010	10	
	00000000001000000	00000000010100000	1		01110011111001011	12	
	00000000010000000	00000000101000000	1		11100111110010000	10	
	00000000100000000	00000001010000000	1		11001111010000010	11	
	00000001000000000	00000010100000000	1		10011110100001100	11	
	00000010000000000	00000101000000000	1		01111010000101011	12	
	00000100000000000	00001010000000000	1		11110100001011010	13	
	00001000000000000	00010100000000000	1		11110100001110000	13	
	00010000000000000	00101000000000000	1		11101000011100001	14	
	00100000000000000	01010000000000000	1		11010000111000000	13	
	01000000000000000	10100000000000000	1		10100001110000001	13	
	10000000000000000	01000000000000000	1		01000011100000010	13	
	00000000000000001	10000000000000000	1		10000111000000100	12	
	00000000000000010	00000000000000000	1		00001110000001000	13	
	00000000000000100	00000000000001010	1		00011100000010000	12	
	00000000000001000	00000000000010100	1		00111000000100000	10	
	00000000000010000	00000000000101000	1		01110000001000000	12	
	00000000000100000	00000000001010000	1		11100000010000000	10	
	00000000001000000	00000000010100000	1		11000000100000000	8	
	00000000010000000	00000000101000000	1		10000001000000000	9	
	00000000100000000	00000001010000000	1		00000010000000000	8	
	00000001000000000	00000010100000000	1		00000100000000000	9	
	00000010000000000	00000101000000000	1		00001000000000000	6	
	00000100000000000	00001010000000000	1		00010000000000000	6	
	00001000000000000	00010100000000000	1		00100000000000000	9	
	00010000000000000	00101000000000000	1		01000000000000000	8	
	00100000000000000	01010000000000000	1		10000000000000000	9	
	01000000000000000	10100000000000000	1		00000000000000000	12	
	10000000000000000	01000000000000000	0		00000000000000000	8	

9. REFERENCES

[1] P.K. Sinha, *Distributed Operating Systems Concepts and Design*, Prentice-Hall of India Private Ltd., March 2002

[2] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems Concepts and Design*, Pearson Education, 2003

[3] Ernest Chang and Rosemary Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes," *Commun.* ACM, vol. 22, pp.281-283, May 1979.

[4] Kalyan Mahata, Meghnath Saha, and Sukanta Das, "Cellular Automata Based Coordinator Selection Scheme in Distributed System," *in Proceedings of CSC, 2009*, pp. 304-310, 2009