MODIFIED BULLY ELECTION ALGORITHM IN DISTRIBUTED SYSTEMS

M. S. Kordafshari, M. Gholipour, M.Jahanshahi, A.T. Haghighat,

Department of Electrical, Computer & IT, Islamic Azad University, Qazvin Branch,
Qazvin, Iran
Atomic Energy Organization of Iran (AEOI), NPPD, Tehran, Iran

Abstract:

Leader election is an important problem in distributed computing, and it is applied in many scientific fields such as communication network [1,2,3,4,5], centralized mutual exclusion algorithm [6,7], centralized control IPC, Berkeley algorithm, etc. Synchronization between processes often requires one process acting as a coordinator. The coordinator might not remain the same, because might get crashed. Bully election algorithm is one of the classic methods which is used to determine the process with highest priority number as the coordinator. In this paper, we will discuss the drawbacks of Garcia_Molina's Bully algorithm and then we will present an optimized method for the Bully algorithm called modified bully algorithm. Our analytical simulation shows that, our algorithm is more efficient rather than the Bully algorithm with fewer message passing and fewer stages.

Key-word:

Bully algorithm, modified Bully algorithm, election, distributed systems, Message passing, coordinator

1. Introduction

Election of a leader is a fundamental problem in distributed computing. It has been the subject of intensive research since its importance was first articulated by Gerard Le Lann [8] in 1977. The practical importance of elections in a distributed computing is further emphasized by Garcia_Molina's Bully algorithm [9] in 1982. Based on a network topology, to elect a high-priority leader, many kinds of leader election algorithm have been presented. There are some algorithms about the election such as Fredrickson and Lynch [10], Singh and Kurose [11], etc. Other algorithms such as B.Awerbuch algorithm [12], Gallage & Humblet algorithm [13], Gafini algorithm [14], Chin & Ting Algorithm [15], Chow & Luo Algorithm [16] are based on spanning tree. Also some related papers proposed based on Ring Algorithm for

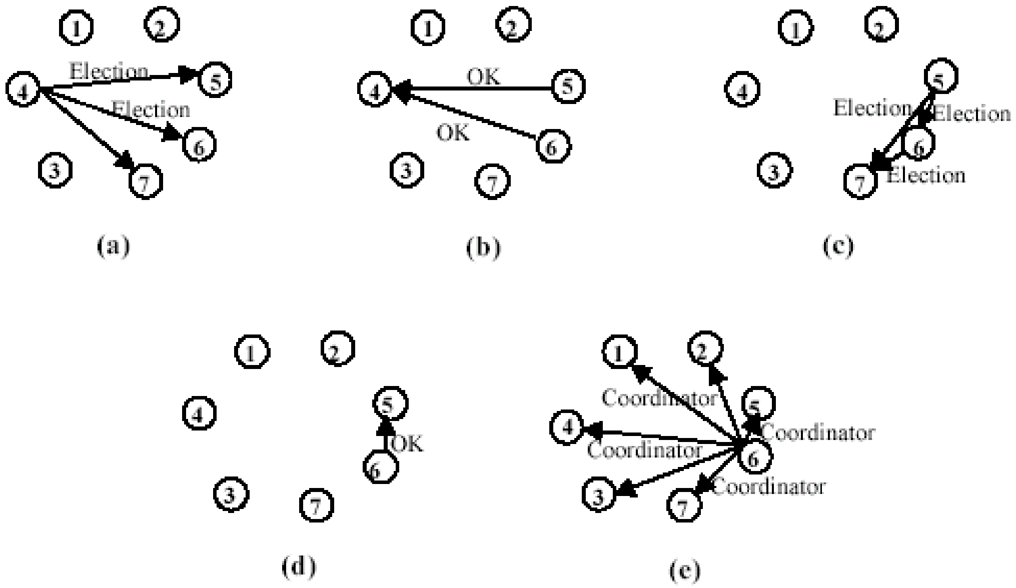


Figure1.The Bully election algorithm.(a) process 4 holds an election . (b) Processes 5 and 6 respond, telling 4 to stop. (c) Now 5 and 6 each hold an election. (d) Process 6 tells 5 to stop. (e) Process 6 wins and tell everyone

electing the leader such as Change-Roberts [17], Peterson [18], Franklin [19], Hishberg [20], etc.

For electing a leader many models of synchronous and asynchronous computation are described in [21,22]. In asynchronous communication there is a delay in the transmission of messages [23]. Bully algorithm is one of the most applicable elections Algorithms that was presented by Garcia_Molina in 1982.

In this paper, we discuss the drawback of synchronous Garcia_Molina's Bully algorithm and modify it with an optimal message algorithm. We show that our algorithm is more efficient than Garcia_Molina's Bully algorithm, because of fewer message passing and fewer stages.

In future work we will implement our algorithm with asynchronous model in order to decrease number of message passing in asynchronous bully algorithm .The rest of paper is organized as follows: In section 2 Garcia_Molina's Bully algorithm is briefly introduced and its advantage and disadvantage are discussed. In section3 improved method for solving Bully algorithm drawbacks is presented. In section 4 Garcia_Molina's bully algorithm and our modified algorithm are compared. In the section 5 we conclude these algorithms .Finally in the last section we explain future work.

2. Bully Algorithm

Bully algorithm is one of the most applicable election Algorithms which was presented by Garcia_Molina in 1982. In this algorithm each process has a unique number to distinguish them and each process knows other's process number. In this algorithm processes don't know which ones are currently up and down. The aim of election Algorithm execution is selecting one process as leader (Coordinator) that all processes agree with it. (I.e. process with the highest id number).

Suppose that the process P finds out the coordinator crashed. This algorithm has the following steps: (As figure 1)

Step1- when a process, P, notices that the coordinator crashed, it initiates an election algorithm

1.1-P sends an ELECTION message to all processes with higher numbers respect to it.

1.2- If no one responses, P wins the election and becomes a coordinator.

Step2- when a process receives an ELECTION message from one of the processes with lower numbered response to it:

2.1- The receiver sends an OK message back to the sender to indicate that it is alive and will take over.

2.2- The receiver holds an election, unless it is already holding one.

2.3- Finally, all processes give up except one that is the new coordinator.

2.4- The new coordinator announces its victory by sending a message to all processes telling them, it is the new coordinator.

Step3- immediately after the process with higher number compare to coordinator is up, bully algorithm is run.

The main drawback of Bully algorithm is the high number of message passing .As it is mentioned before the message passing has order $o(n^2)$ that increases traffic in network.

The advantages of Bully algorithm are that this algorithm is a distributed method with simple implementation. This method requires at most five stages, and the probability of detecting a crashed process during the execution of algorithm is lowered in contrast to other algorithms. Therefor other algorithms impose heavy traffic in the network in contrast to Bully algorithm. Another advantage of this algorithm is that only the processes with higher priority number respect to the priority number of process that detects the crash coordinator will be involved in election, not all process are involved. In brief, Bully algorithm is a safe way for election, however its traffic is relatively high. In section 3 we proposed a solution to overcome these drawbacks.

3. Modified Bully Algorithm

As has been mentioned in section 2 in Bully algorithm number of messages that should be exchanged between processes is high. Therefore this method imposes heavy traffic in network.

For solving this drawback we will present optimized method by modifying the Bully algorithm, that intensively decreases the number of messages that should be exchanged between processes. Furthermore the number of stages is decreased from at most five stages to at most four stages.

Our algorithm has following steps: (figure 2)

Step1- When process P notices that the coordinator has crashed, it initiates an election algorithm.

Step2- When the process P finds out that the coordinator is crashed, sends ELECTION message to all other processes with higher priority number.

Step3-Each process that receives ELECTION messages (with higher process than P) sends OK message with its unique priority number to process P.

Step4- If no process responses to process P, it will broadcast one COORDINATOR message to all processes, declaring itself as a coordinator. If some process response to process P by comparing the priority numbers, the process P will select the process with the highest priority number as coordinator and then sends to it the GRANT message.

Step5- at this stage the coordinator process will broadcast a message to all other processes and informs itself as a coordinator.

Step6- immediately after the process with higher number compare to coordinator is up, our algorithm is run.

New algorithm not only has all advantages of Bully algorithm also it doesn't has the drawback of Bully algorithm (high number of message passing). Furthermore maximum number of stages is decreased from five stages to four stages.

It is clear that if process P crashes after sending ELECTION message to higher processes, or crashes after receiving the priority numbers from process with higher priority number, higher process wait at most 3D time for coordinator broadcast. (D is average propagation delay), If it will not receive, this process runs the modified algorithm. If a process with higher priority number crashes after sending its priority number to P, process P sends GRANT message to it meaning that it is the highest process and P waits for broadcasting coordinator message. If after D time, process P doesn't receives the COORDINATOR message, it repeats the algorithm again.

Therefore we can use this algorithm as an efficient and safe method to selecting the coordinator.

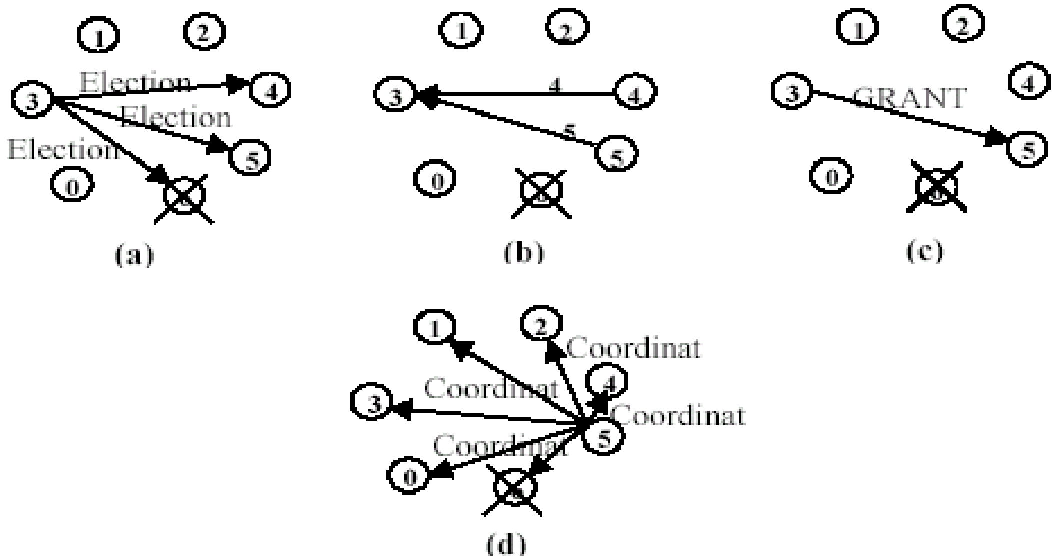


Figure2.The modified Bully election algorithm.(a) process 3 holds an election . (b) processes 4 and 5 and 6 respond, telling its unique priority number.(c) Now 3 comparing the priority number and selects the highest process(process 5) and sends a message to it(GRANT) (d) process 5 tell to everyone that "it is coordinator"

3.1 A novel solution for a drawback of Bully algorithm

In Bully algorithm when more than one process or all processes find out that the coordinator has crashed simultaneously, all of them run in parallel Bully algorithm, therefore heavy traffic imposed to the network.

For solving this problem in modified bully algorithm we act as follow (figure3).

Step1-When process P realizes that the coordinator has crashed, it initiates modified bully election algorithm presented in section 3.

Step2-When process P' (P' may be P) receives the ELECTION message from process or processes with lower priority number compare to itself, it waits a short time that can be specified perfectly and then answers to the process with lowest priority number only. In this situation if P=P' (This process initiates the algorithm and also received the ELECTION message from other processes), then stops the algorithm.

Step3-After process P' answered to P, if P' receives an ELECTION message from process R(R<P<P'), P' answers to process R by sending its priority number and sends STOP message to process P.

Step4 *- when a* process receives the STOP message stops the algorithm immediately.

Step5- if process p neither receives any response from other process(es),nor does it receive any ELECTION message form processes with lower Priority number , then in this case it can inform other processes containing it(P) as COORDINATOR.

The pseudo code for this algorithm is represented in figure 4.

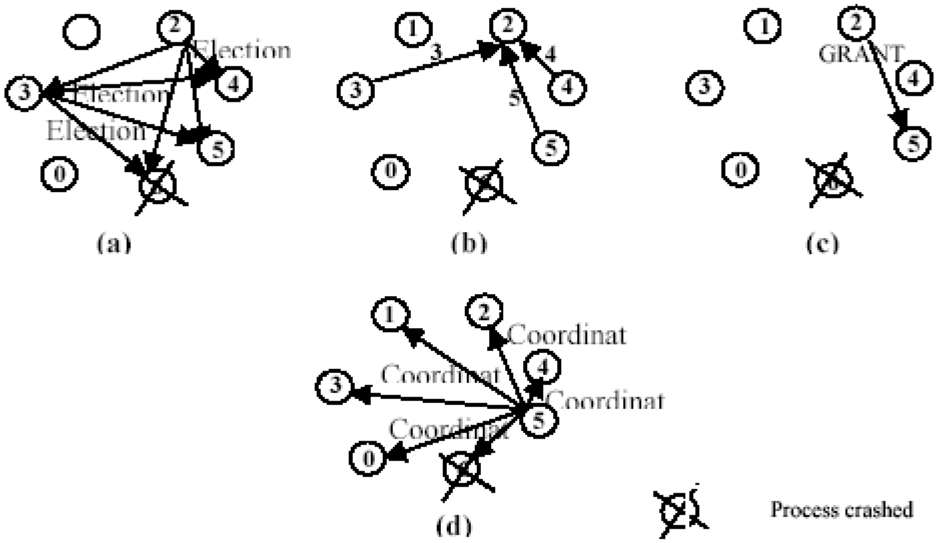


Figure3.The modified Bully election algorithm.(a) process 3 and 2 find out the crashed coordinator simultaneously and therefore each of which send ELECTION messages separately. (b)other processes that receive more than one ELECTION message should only send their own priority number to process with lowest id number (in this case 2) .(c,d) process 3 stop the algorithm because receiving the ELECTION message from process 2 .process 2 continue the algorithm.

4. Advantages of our algorithm in contrast with bully Algorithm

In this section we will compare Bully algorithm and modified bully algorithm:

In point of number of stages:

In point of number of stages Bully algorithm always is executed in five stages, while our algorithm find out the coordinator after four stages.

4.1 Analytical comparison of two algorithms if only one process detects the crashed coordinator

If only one process detects crashed coordinator

n: The number of processes

r: The priority number of processes that find out the crashed coordinator

$N_{(r)}$: The number of messages passing between processes when the r-th member detects the crashed Coordinator.

In bully modified algorithm the number of massages passing between processes for performing election is obtained from the following formula:

$$N_{(r)} = 2*(n-r)+n \qquad \mathbf{(1)}$$

Which has Order $o(n)$. In the worst case that is $r=1$ (process with lowest priority number finds out crashed coordinator):

$$N_{(1)} = 2*(n-1)+n = 3n-1 \qquad (2)$$

Whereas the number of massage passing between processes in the Bully algorithm for performing election is obtained from the following formula:

$$N_{(r)} = (n-r+1)(n-r)+n-1 \qquad (3)$$

In the worst case that is $r=1$ (process with lowest priority number detects crashed coordinator):

$$N_{(1)} = n^2-1 \qquad (4)$$

Which has Order $O(n^2)$

Number of messages in modified bully algorithm will be equal to $3n-1$ that obviously means this modified algorithm is better than bully algorithm with fewer messages passing and the fewer stages.

Figure 5 clearly shows the comparison between bully algorithm and modified bully algorithm (when one process finds out that crashed coordinator). Horizontal axis indicates the priority number of processes that find out crashed coordinator, and vertical axis indicates the number of message passing. For example if the number of processes is 1000 and 100th process finds out that crashed coordinator, in bully algorithm the number of message passing is equal 811899 but the number of message passing in modified bully algorithm is equal 2800.

4.2 Analytical comparison of two algorithms if set of $S=\{r_1, r_2, \ldots, r_m\}$ run the algorithm simultaneously.

Now assume that the set of processes in $S=\{r_1, r_2, \ldots, r_m\}$ from processes find out the crashed coordinator concurrently (r_1 is lowest process):

In bully algorithm the number of message passing between processes for performing election is obtained from the following formula:

$$T = (n-r_1+1)(n-r_1)+n-1 \qquad (5)$$

In our modified algorithm the number of message passing between processes for performing election is obtained from the following formula:

$$T = (n-r_1) + \sum_{\{r_j | r_j \in S\}} (n-r_j) + n \qquad (6)$$

In bully algorithm the number of message passing is based on the process with lowest priority number. That means there isn't any difference between state that only process r_1 detects the crashed coordinator and state that in witch the set of $S = \{r_1, r_2, ..., r_m\}$ detects crashed coordinator.

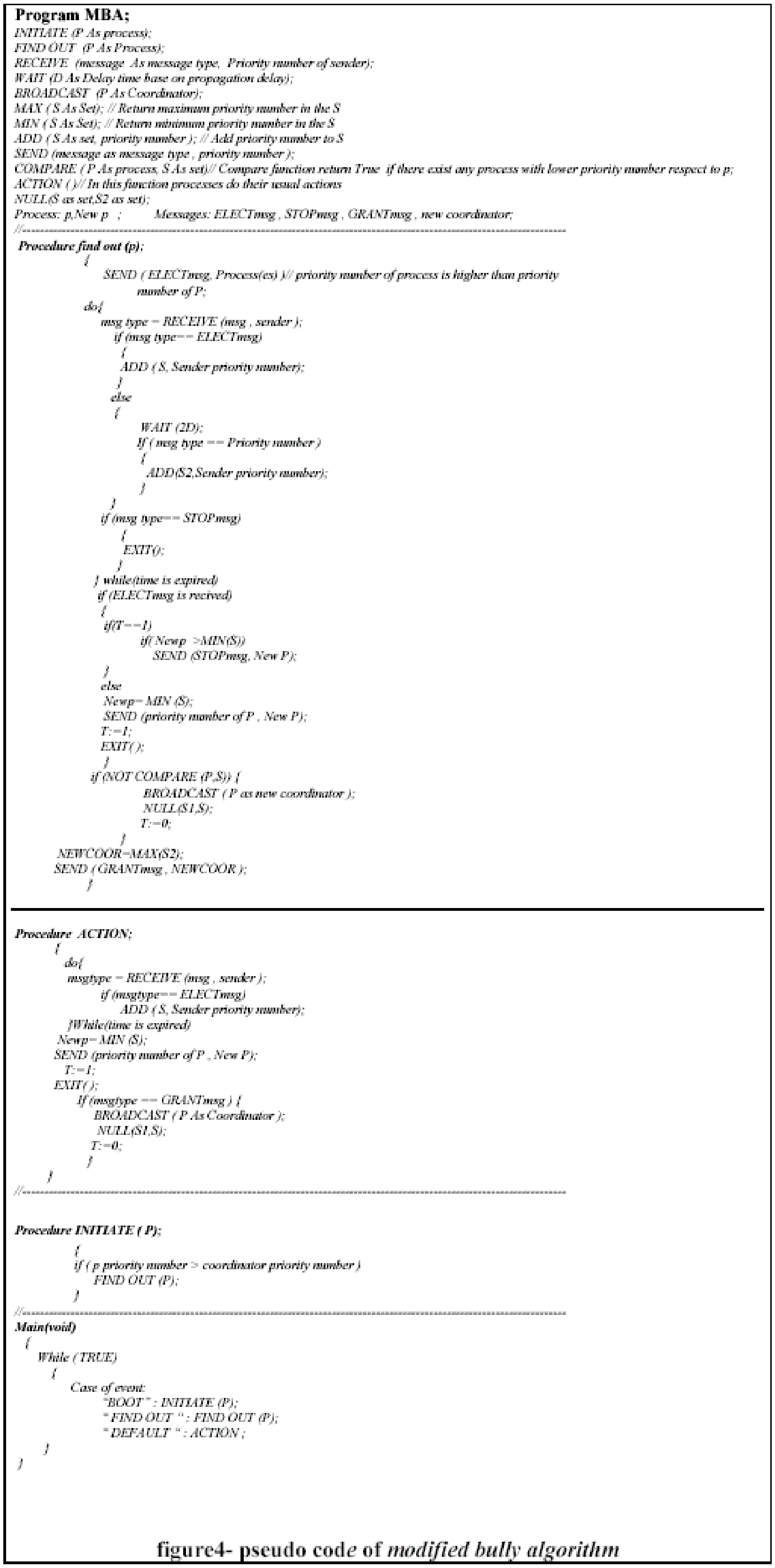
But in modified algorithm set of $S = \{r_1, r_2, ..., r_m\}$ are also important. If the priority numbers of the processes that detects the crashed coordinator is higher, the number of message passing will be decreased considerably.

5.Conclusion

In this paper, we discussed the drawbacks of Garcia_Molina's Bully algorithm and then we presented an optimized method for the Bully algorithm called modified bully algorithm. Our analytical simulation shows that our algorithm is more efficient rather than the Bully algorithm, in both number of message passing and the number of stages, and when only one process run the algorithm message passing complexity decreased from $O(n^2)$ to $O(n)$ (formula 1,3).In this analysis we consider the worst case in modified algorithm. Result of this analysis clearly shows that modified algorithm is better than bully algorithm with fewer message passing and the fewer stages.

6. Future work

In future work we will implement our algorithm with asynchronous model in order to decrease number of message passing in asynchronous bully algorithm.



```
Program MBA;
INITIATE (P As process);
FIND OUT  (P As Process);
RECEIVE  (message  As message type,  Priority number of sender);
WAIT (D As Delay time base on propagation delay);
BROADCAST  (P As Coordinator);
MAX ( S As Set); // Return maximum priority number in the S
MIN ( S As Set); // Return minimum priority number in the S
ADD ( S As set, priority number ); // Add priority number to S
SEND (message as message type , priority number );
COMPARE ( P As process, S As set)// Compare function return True  if there exist any process with lower priority number respect to p;
ACTION ( )// In this function processes do their usual actions
NULL(S as set,S2 as set);
Process: p,New p   ;         Messages: ELECTmsg , STOPmsg , GRANTmsg , new coordinator;
//-----------------------------------------------------------------------------------------------------------
 Procedure find out (p);
                {
                     SEND ( ELECTmsg, Process(es) )// priority number of process is higher than priority
                              number of P;
                do{
                    msg type = RECEIVE (msg , sender );
                       if (msg type== ELECTmsg)
                         {
                         ADD ( S, Sender priority number);
                         }
                      else
                       {
                              WAIT (2D);
                             If ( msg type == Priority number )
                              {
                                ADD(S2,Sender priority number);
                              }
                      }
                    if (msg type== STOPmsg)
                         {
                          EXIT();
                         }
                  } while(time is expired)
                   if (ELECTmsg is recived)
                    {
                    if(T==1)
                              if( Newp  >MIN(S))
                                 SEND (STOPmsg, New P);
                    }
                    else
                     Newp= MIN (S);
                     SEND (priority number of P , New P);
                    T:=1;
                    EXIT( );
                     }
                 if (NOT COMPARE (P,S)) {
                             BROADCAST  ( P as new coordinator );
                             NULL(S1,S);
                            T:=0;
                        }
      NEWCOOR=MAX(S2);
     SEND ( GRANTmsg , NEWCOOR );
            }

Procedure  ACTION;
         {
           do{
            msgtype = RECEIVE (msg , sender );
                   if (msgtype== ELECTmsg)
                        ADD ( S, Sender priority number);
            }While(time is expired)
          Newp= MIN (S);
         SEND (priority number of P , New P);
           T:=1;
         EXIT( );
              If (msgtype == GRANTmsg ) {
                  BROADCAST ( P As Coordinator );
                   NULL(S1,S);
                 T:=0;
                 }
        }
//-----------------------------------------------------------------------------------------------------------

Procedure INITIATE ( P);
              {
              if ( p priority number > coordinator priority number )
                  FIND OUT (P);
              }
//-----------------------------------------------------------------------------------------------------------
Main(void)
  {
    While ( TRUE)
       {
          Case of event:
                 "BOOT" : INITIATE (P);
                 " FIND OUT " : FIND OUT (P);
                 " DEFAULT " : ACTION ;
      }
 }
```

figure4- pseudo code of *modified bully algorithm*

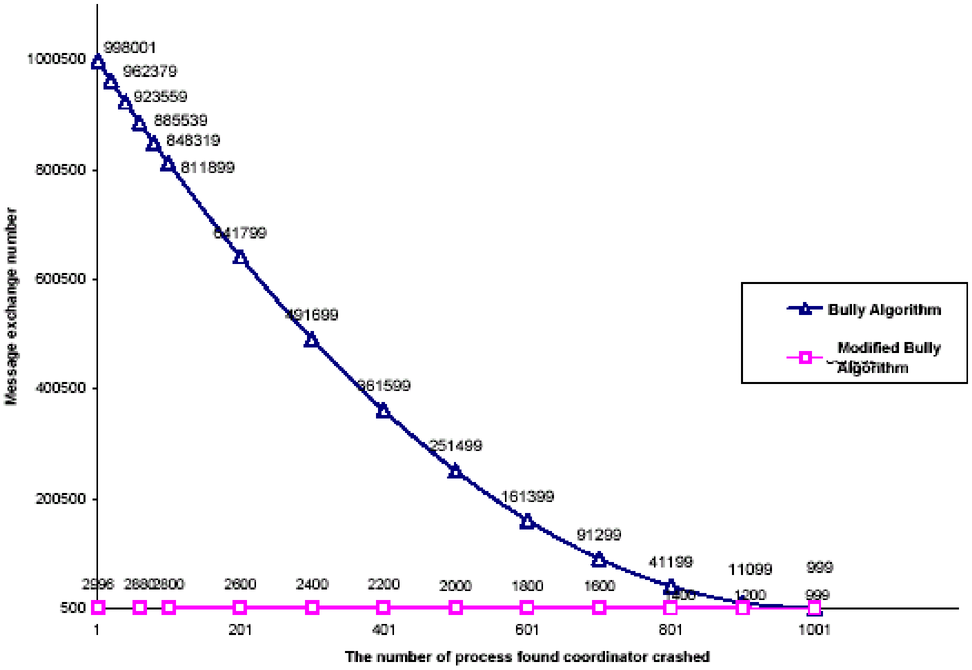


Figure5-The comparison of two algorithms if only one process finds out crashed coordinator (n=1000)

7. References

[1]Sung-Hoon-Park, Yoon Kim, And Jeoung Sun Hwang " An Efficient Algorithm for Leader-Election in Synchrous Distributed Systems." IEEE Transaction on Computers, vol. 43, no. 7, pp.1991-1994, 1999.

[2] H. Abu-Amar and J. Lokre " Election In Asyncrouns Complete Network With Intermitted Link Failures." IEEE Transaction on Computers, vol. 43, no. 7, pp.778-788, 1994.

[3] H.M. Sayeed, M. Abu-Amara, and H. Abu-Avara, "Optimal Asynchronous Agreement and Leader Election Algorithm for Complete Networks with Byzantine Faulty Links." Distributed Computing, vol.9, no.3, pp.147-156, 1995.

[4] J. Brunkreef, J.P. Katoen, and S. Mauw, " Design and Analysis of Dynamic Leader Election Protocols in Broadcast Network, "Distributed Computing, vol.9, no.4, pp.157-171. 1996

[5] G. Singh, "Leader Election in the Presence of Link Failures," IEEE Transaction on Parallel and Distributed Systems, vol. 7, no. 3, pp.231-236, March 1996.

[6] D.Menasce R.Muntz and J.Popek," A locking protocol for resource coordination in Distributed databases "

ACM TODS , VOL.5,NO.2,pp.103-138, JUNE 1980.

[7] P.A Alsberg and J.Day “ A principle for resilient sharing of Distributed Resource “ in Proc.2nd Inte. Conf , on software Engg.,(Sanfrancisco,Oct.1976).

[8] G. Le Lan,”Distributed System – Towards a Formal Approach, ” In Information Processing 77,B. Gilchrist,Ed.Amsterdam,The netherlands:North-Holland, pp.155-160.1977

[9] H. Garcia-Molina, “Elections in Distributed Computing System,” IEEE Transaction Comput, Vol.C-31,pp.48-59,Jan.1982.

[10] G.Fredrickson and N.Lynch, “ Electing a leader in a synchronous Ring ", JACM, Vol 34,NO.pp-199-115 (JAN 1987).

[11] Singh, S., Kurose, J.F., “Electing ‘good’ leaders (election leader algorithm),” Journal of Parallel and Distributed Computing, Vol. 21, No. 2, pp. 184-201. (May 1994)

[12] CHANG E.,ROBERTS R. “ An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of processes.” Comm.of the ACM 22:5,pp.281-283, 1979

[13] R. Gallager, P. Humblet, and P.Spira, “ A Distributed Algorithm for Minimum Weighted Spanning tree.” ACM trans.On programing Language and Systems. 5(1):pp-77, 1983.

[14] G. Gafini. “Improvement in time complexity of two message optimal algorithm.” Proc. Princip;es of Distributed Computing Conf., pp- 175-183, 1995.

[15] F. Chin and H. F. Ting. “ An Almost Linear time and EMBED Equation.3 Message Distributed Algorithm for Minimum Weighted Spanning Tree. Proc. Foundation of Computer Science Conf., pp-257-266, 1995.

[16] R. Chow,K. Luo, and R. Newman-Wolf. “ An Optimal Distributed Algorithm for Failure Driven Leader Election in Bounded degree Network.” Proc IEEE Workshop on Future Tends of Distributed Computing Systems, pp. 136-141, 1996

[17] PETERSON G.L “An O(nlogn) Unidirectional Algorithm for Circular Extrema problem . ACM Transaction on Programming Languages and Systems”4:4,pp.758-762.1982

[18] Wm. Randolph Franklin, "On an Improved Algorithm for Decentralized Extrema Finding in Circular Configurations of Processors," CACM, page 336-337. May 1982.

[19] B.Awerbuch, “ Optimal Distributed Algorithm for Minimum weight spanning tree, Leader Election and related problems” , ACM STOC, pp.230-240, 1987

[20] D.S. Hirshberg and J.B. Sinclair. “ Decentralized Extrim Finding in Circular Configurations of Processors.” Communications of ACM, 23(11) pp-627-629, 1980 .

[21] M. fischer, N. Lynch, and M. Paterson, “Impossiblity of Distributed Consensus with one Faulty Process,” Journal of the ACM, pp.374-382.(32) 1985.

[22] Tushar Deeppak Chandra and Sam Toueg. “ Unreliable Failure Detectors for Relaiable Distributed Systems” Journal of ACM, Vol,43 No.2, pp-225,267,March 1996 .

[23]Sung-Hoon-Park,“A Probablistically Correct Election Protocol in Asynchronous Distributed System “, APPT, LNCS 2834, pp.177-183, 2003.