2012 International Conference on Information and Network Technology (ICINT 2012)
IPCSIT vol. 37 (2012) © (2012) IACSIT Press, Singapore

Achieving Consensus in The Presence of Both Link and Node Failure

Monalisa Dey [+], Prasenjit Dey and Mamata Dalui

Department of Computer Science and Engineering, National Institute of Technology, Durgapur, WB, India 713209

Abstract. This work reports a simple and efficient solution to the consensus problem among the nodes of a distributed system. The solution proposed considers both the communication link and node failures thereby making the fault coverage better than the schemes reported so far. Detection of the faulty links and nodes make this scheme more transparent. Better efficiency is achieved through the early disposal of faulty nodes. This solution uses two rounds of message exchanges.

Keywords: byzantine agreement, consensus, early disposal, malicious and dormant link faults, malicious and dormant node faults, reaching agreement.

1. Introduction

In a distributed system, it is sometime necessary for all the nodes to agree on a common value and hence reach a unanimous decision. The process of reaching an agreement is easy when all the components (nodes and links) in the system are non-faulty. But it becomes more complicated whenever some of the components are faulty. In order to achieve agreement still tolerating these faulty components, certain agreement protocols have been proposed in the literature. Lamport [1] had studied such a problem and it is called the Byzantine Agreement [1]. Another related problem, the consensus problem has also been studied extensively in [2], [3], [5]. In byzantine agreement, one node (source) sends its initial value to all others. If the source is non-faulty, all the non-faulty nodes agree on that value. In the consensus problem, however, each of the nodes has their own initial value. They exchange these values among themselves and finally all the non-faulty nodes agree on the same value.

In this paper, we have addressed the consensus problem. Most of the solutions to consensus problem have addressed either the assumption on node failure [1], [3], [4], [6], [7] or link failure [2], [5]. However none of them have considered both failures simultaneously. This encouraged us to propose a solution MFC (Maximum Fault Coverage) to the consensus problem assuming both link and node failure. The solution is efficient in terms of minimum number of rounds required to reach consensus and detection of the faulty components. The symptoms of a faulty link and node can be classified into two types: - dormant (crash) and malicious (arbitrary).

Yan and Chin [2] had found out a solution to the consensus problem by protocol FLINK, but they have treated all link failures as malicious. Wang and Yan [5] had solved this problem by protocol DLFM but they have dealt with only dormant and malicious link failures. Also these protocols are unable to detect any of the faulty components. We have not only considered the unreliability of both the communication links and the nodes, but have also devised a method to detect the faulty components. DLFM can tolerate $m \leq \lceil (n - d - 3)/2 \rceil$ malicious and dormant faulty links only, FLINK can tolerate $\lfloor (n/2) \rfloor - 1$ malicious faulty links only. MFC can tolerate t node faults with $n > 2t$, where n is the total number of nodes in the distributed system. It

[+] Corresponding author. Tel.: + 91-9804575406.
E-mail address: monalisa.dey.21@gmail.com.

can simultaneously tolerate m malicious and d dormant link faults respectively, in a fully connected network to reach consensus where $m \leq \lceil((n-t)-d-3)/2\rceil$. Just two rounds of message exchanges are required for achieving consensus. The amount of message exchange required is $O(n^2)$.

2. The Proposed Model

In a large distributed network reaching a common decision among the nodes require huge information exchange. If the faulty nodes are identified early the whole process can be speeded up. A link may also be damaged. The proposed model takes care of both dormant and malicious faults in nodes and links and the system can reach a quick consensus in just two rounds of message exchanges.

In the first round, all the nodes exchange their own initial values. Each node i then arranges these values in a vector v_i. In the second round each node sends their vector v_i to all other nodes, a 2d array MAT_i is constructed at each node i. MAT_i is constructed using vector v_j as the j^{th} column in the array. In a faulty system the arrays created are not the same. A node, by inspecting its MAT can identify the dormant and malicious nodes and links. Any node detects all the faulty nodes by discarding the corresponding rows and columns from its MAT. Each node keeps count of the faulty nodes it detects. If a node sees that the number of valid rows in its array is greater than the number of faulty nodes, it decides on majority of 0's or 1's in the row thereby forming majority matrix MAJ. By inspecting their individual MAJ each node then agrees on a common value thereby reaching consensus. The common value is nothing but the majority of the 0's and 1's in the majority matrix MAJ for each node. If a node has the same number of 0's and 1's it agrees on a common value 0.

In this section the proposed protocol MFC is formally presented. The following assumptions are taken:

1) F_d and L_d are the number of dormant faulty nodes and links respectively.

2) F_m and L_m are the number of malicious faulty nodes and links respectively.

3) A faulty node doesn't alter the values it receives from others.

4) $f = F_d + F_m$ is the total number of faulty nodes.

5) Links are bidirectional in nature and the nodes on either end of a malicious faulty link, cannot be malicious faulty.

Algorithm 1:

First round: Each node sends its initial value to all. If any node i does not receive any value from node j, update $v_i[j] = M$.

Second round: All nodes send their vectors to all other nodes. If a node i doesn't receive any values from node j, update $MAT_i[k][j] = M$. The detailed algorithm is presented in Figure 1.

The following example illustrates the precise steps of our proposed scheme MFC in algorithm 1. Let us consider a system of seven nodes. Initial values $v_i=0$ for i=1, 2 and 5. And $v_i=1$ for i=3, 4, 6 and 7. The vectors received by the nodes after the 1st round is shown in Figure 2. The 2d arrays formed at each node after the 2nd round of message exchange are shown in Figure 3. Nodes p_2 and p_3 are taken to be malicious and dormant faulty nodes respectively. $Link_{24}$ (link between node 2 and node 4) and $link_{57}$ are assumed to be dormant and malicious faulty links respectively. In Figure 4 the step by step working of the next stage is shown as described in the proposed scheme. Figure 4(a) shows the 2d array constructed at node 4. In the next step shown in Figure 4(b) we have replaced the values of the row and column entries of malicious node 2 with any numerical value other than 0 and 1, say X to dispose of faulty nodes. Next we construct the majority matrix MAJ of node 4 by taking majority of each row separately, ignoring the row and column entries corresponding to the faulty nodes 2 and 3 shown in Figure 4(c). Each row k of the matrix MAJ is inspected to find out the majority value which then becomes the decision value for node 4. In Figure 4(c) the majority value in MAJ is 1 so the decision value for node 4 is 1. Similarly, $DEC_i = 1$ □ i (excluding 2, 3).

The following lemmas and theorems are used to prove the correctness of MFC as in [5].

Lemma 1: Let node i has an initial value v_i, irrespective of whether $link_{ij}$ is perfect or dormant MAJ_i in MAT_j should be v_i.

```
1: initialize f ← 0;
2: for i ← 1 to n do
3:   initialize c←n;
4:   for k ← 1 to n do
5:     for j ← 1 to n do
6:       if MAT_i[k][j]= =M then
7:         for x ← 1 to n do
8:           if MAT_i[j][x]!=M then
9:             decrement c.
10:            if MAT_i[k][j] = = MAT_i[j][k] then
11:              link_kj is dormant link faulty.
12:            end if
13:          end if
14:        end for
15:      end if
16:      if c= =n and k= =1 then
17:       report node j is dormant faulty and increase the
          count of f.
18:       end if
19:      end for
20:   end for
21:   for k ← 1 to n do
22:     for j ← 1 to n do
23:       if MAT_i[k][j] != v_k then
24:       if MAT_i[j][k] != v_j then
25:         report link_kj  is malicious faulty
26:       else
27:         report node k is malicious faulty and increase the
            count of f.
28:         set all entries of row k and column k as any
            numerical value say X.
29:         break out from inner loop.
30:       end if
31:     end if
32:    end for
33:   end for
      Number of non-faulty nodes NFN=n-f
34: end for
35: if  NFN > f then
36:   for i ← 1 to n do
37:     for k ← 1 to n do
38:       ignore row k if all its column are either M or X else
          take the majority of 0/1 of each row k of MAT_i and store
          it in a majority matrix MAJ_i
39:     end for
40:   end for
41: end if
    ( non-faulty node i do)
42: Each row k of MAJ_i  is inspected to find out the
    majority between 0 and 1 values and finally that
    becomes the decision value for node i.
```

Fig. 1: Algorithm 1

Proof: **Case 1**: Suppose if $link_{ij}$ is perfect, the node j will receive v_i from node i in the first round and $v_{ij} = v_i$ in MAT_j. The value v_i of node i will be broadcasted to the other nodes. There are at most $\lceil ((n-t)-d-3)/2 \rceil$ malicious faulty links in the system. Now, in the second round, node j receives at least $(n-t-d-1) - \lceil ((n-t)-d-3)/2 \rceil = \lfloor (n-t-d+1)/2 \rfloor$ v_i's in the i-th row of MAT_j , where d is the number of dormant link faults which will be eliminated during the voting of majority. Hence, there are at least $\lfloor (n - t - d + 1)/2 \rfloor$ v_i's in the i^{th} row, and the majority value in the i^{th} row should be equal to v_i.

Case 2-1: $Link_{ij}$ is dormant and n is an odd number, the node j will receive M from node i in the first round and v_{ij}=M in MAT_j. Meanwhile, the value v_i of node i will be broadcasted to the other nodes. There are at most $\lceil ((n - t) - d - 3)/2 \rceil$ malicious links and d dormant links in the system. After the second round, node j receives at least (d+1) M's and at least n-t-(d+1) - $\lceil ((n-t)-d-3)/2 \rceil = \lfloor (n-t-d+1)/2 \rfloor$ v_i's in the i^{th} row of MAT_j , where d is the number of M which will be eliminated during the voting of majority. Hence, there are

P_1	P_2	P_3	P_4	P_5	P_6	P_7
0	**0**	**0**	**0**	**0**	**0**	**0**
1	**0**	**0**	**M**	**1**	**0**	**1**
M	**M**	**M**	**M**	**M**	**M**	**M**
1	**M**	**1**	**1**	**1**	**1**	**1**
0	**0**	**0**	**0**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**0**	**1**	**1**

Fig. 2: Vectors received by each node after first round

$$MAT_1=\begin{vmatrix}0&0&M&0&0&0&0\\1&0&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&1&1&1&1\\0&0&M&0&0&0&1\\1&1&M&1&1&1&1\\1&1&M&1&0&1&1\end{vmatrix}\quad MAT_2=\begin{vmatrix}0&0&M&M&0&0&0\\1&0&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&M&1&1&1\\0&0&M&M&0&0&1\\1&1&M&M&1&1&1\\1&1&M&M&0&1&1\end{vmatrix}\quad MAT_3=\begin{vmatrix}0&0&M&0&0&0&0\\1&0&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&1&1&1&1\\0&0&M&0&0&0&1\\1&1&M&1&1&1&1\\1&1&M&1&0&1&1\end{vmatrix}\quad MAT_4=\begin{vmatrix}0&M&M&0&0&0&0\\1&M&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&1&1&1&1\\0&M&M&0&0&0&1\\1&M&M&1&1&1&1\\1&M&M&1&0&1&1\end{vmatrix}$$

$$MAT_5=\begin{vmatrix}0&0&M&0&0&0&0\\1&0&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&1&1&1&1\\0&0&M&0&0&0&1\\1&1&M&1&1&1&1\\1&1&M&1&0&1&1\end{vmatrix}\quad MAT_6=\begin{vmatrix}0&0&M&0&0&0&0\\1&0&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&1&1&1&1\\0&0&M&0&0&0&1\\1&1&M&1&1&1&1\\1&1&M&1&0&1&1\end{vmatrix}\quad MAT_7=\begin{vmatrix}0&0&M&0&0&0&0\\1&0&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&1&1&1&1\\0&0&M&0&0&0&1\\1&1&M&1&1&1&1\\1&1&M&1&0&1&1\end{vmatrix}$$

Fig. 3: 2d arrays received by each node after second round

$$MAT_4=\begin{vmatrix}0&M&M&0&0&0&0\\1&M&M&M&1&0&1\\M&M&M&M&M&M&M\\1&M&M&1&1&1&1\\0&M&M&0&0&0&1\\1&M&M&1&1&1&1\\1&M&M&1&0&1&1\end{vmatrix}=\begin{vmatrix}0&X&M&0&0&0&0\\X&X&X&X&X&X&X\\M&X&M&M&M&M&M\\1&X&M&1&1&1&1\\0&X&M&0&0&0&1\\1&X&M&1&1&1&1\\1&X&M&1&0&1&1\end{vmatrix}\quad MAJ\text{ for }MAT_4=\begin{vmatrix}0\\1\\0\\1\\1\end{vmatrix};\ dec_4=1$$

(a) (b) (c)

Fig. 4: Decision Making

n-t-(d+1) non-M's and at least ⌊ (n−d+1)/2⌋ (greater than ⌈ ((n−t)−(d+1)+1)/2⌉ = ⌈ ((n−t)−d)/2⌉the majority required when n is in odd) v_i's in the i^{th} row, so, the majority value in the i^{th} row should be equal to v_i.

Case 2-2: $Link_{ij}$ is dormant and n is an even number, the node j will receive M from node i in the first round and v_{ij}=M in MAT_j. Meanwhile, the value v_i of node i will be broadcasted to the other nodes. There are at most ⌈ ((n − t) − d − 3)/2⌉ − 1 malicious links and d dormant links in the system as if d ≥ 1 and n is in even. After the second round, node j receives at least (t+d+1) M's and at least (n − (t + d + 1)) − (⌈ ((n − t) − d − 3)/2⌉ − 1) = ⌊ ((n − t) − d + 1)/2⌋ + 1 v_i's in the i^{th} row of MAT_j, where d is the number of M which will be eliminated during the voting of majority. Hence, there are n-(t+d+1) non-M's and at least ⌊ ((n − t) − d + 1)/2 ⌋ + 1 (greater than ⌈ (n − (t + d + 1) + 1)/2⌉ = ⌊ ((n − t) − d)/2⌋ the majority required when n is in even) v_i's in the i^{th} row, so, the majority value in the i^{th} row should be equal to v_i.

Theorem 1: The decision taken by each node i are correct.

Proof: For each node i the majority between the 0 and 1 entries in each row of its 2d array MAT_i is computed and the majority matrix MAJ_i is constructed using those values. The faulty nodes do not take part in constructing MAJ_i. Each row k in array MAT_i is nothing but the values that node k sends to each node i. Since the total number of malicious links in an n node system containing d dormant link faults can be maximum ⌈ ((n − t) − d − 3)/2⌉, the maximum number of faulty values in each row cannot be more than ⌈ ((n− t) − d − 3)/2⌉. Thus while calculating the majority value of each row k, the correct initial value of each node k is always obtained. Thus the decision taken by inspecting MAJ it is always correct.

3. Impossibility

In this section, some impossibility of the consensus problem is presented.

Theorem 2: If $link_{ij}$ is malicious faulty, node i or j cannot be malicious faulty nodes.

Proof: Let $link_{ij}$ is malicious faulty and node i is malicious faulty. Node i sends correct values to all other nodes k and wrong value to node j. Due to the link fault ij that value will be further changed and node j will get correct value from node i. Node j on the other hand is perfect, it would send correct value to all except node i. Any other node k while inspecting row i in its 2d array MAT will see that every node has received correct values from node i. Thus, malicious faulty node i will be impossible to detect. Whereas, while inspecting row j, any node k will see that node j has sent its correct value to all except i. Thus node j is wrongly reported as malicious faulty node.

Theorem 3: If the total number of the faulty links tfl > m+d, where m ≤ ⌈ ((n−t)−d−3)/2⌉, achieving consensus is not possible.

Proof: Every node has n-1 links in the system. When tfl > m + d it might happen that a node has more malicious links than perfect links even if the influence of d dormant faults was eliminated. Then even two rounds of message exchange will not be enough to reach consensus as this node will always be confused by the messages transferred through the malicious faulty links it has. And hence it might take a wrong decision.

4. Experimental Results

The fault coverage and tolerance capability of our proposed scheme MFC is compared with DLFM and FLINK in Table 1. The first column represents the total number of participating nodes n. Column 2 (tolerable node faults), column 3 (tolerable malicious links) and column 4 (tolerable dormant links) show the results of MFC. Similarly column 5 and 6 shows the number of malicious and dormant links that DLFM can tolerate. Column 7 and 8 provides the same FLINK can tolerate. The results shown in the table thus points to the fact that the fault coverage capability of our scheme is better than the previous solutions.

5. Performance Study

The performance of MFC is compared with the DLFM [5] and FLINK [2], in terms of the number of messages exchanged to reach an agreement. Table 2 shows the number of message exchanged by DLFM, FLINK and MFC. The 1st column represents the number of participating nodes (n). Two sets of observations for total number of dormant nodes and dormant links are taken. Column 2 (tolerable dormant links), column 3 (tolerable malicious links), column 4 (tolerable dormant nodes) and column 5 (total no. of messages exchanged) give the results of MFC. Column 6 (dormant link faults), column 7 (malicious link faults) and

column 8 (messages exchanged) show the results of DLFM. Column 9 (malicious link faults) and column 10 (messages exchanged) show the results of FLINK. Figure 5 and Figure 6 shows the performance comparisons between these three protocols in term of number of messages exchanged. The results shown in Table I and II point to the fact that the proposed solution maximizes the fault coverage capability of a system. As the size of the network and the number of dormant faults increase, the total number of messages exchanged is reduced and hence efficiency of the proposed scheme increases.

6. Conclusion

In a distributed system, in reality, both the nodes and links of a fully connected network can be faulty. These faults can again be categorized into dormant and malicious. MFC mainly deals with the issue of reaching consensus in a fully connected distributed system where both the nodes and the communication links can be faulty. It ensures a correct decision with just two rounds of message exchanges. A better fault coverage is obtained than the state-of-art solutions. Faulty nodes are disposed of early leading to a quicker solution. Detection of the faulty components makes this scheme more efficient. We will also see that the number of messages exchanged to reach consensus is lesser than the previous solutions and as the number of dormant faults increase, the efficiency of our scheme increases. However further work can be done on this scheme to reduce the information exchange complexity.

Table. 1: Fault coverage capability of MFC, DLFM and FLINK

n	MFC			DLFM		FLINK	
	#t	#m	#dl	#m	#dl	#m	#d
4	0	0	<=2	0	<=2	0	0
		1	0	1	0	1	0
	1	0	1	0	0	0	0
5	0	0	<=3	0	<=3	0	0
		1	<=1	1	<=1	1	0
	1	0	<=2	0	<=2	0	0
		1	0	1	0	1	0
	2	0	1	0	0	0	0
6	0	0	<=4	0	<4	0	0
		1	<=2	1	<=2	1	0
		2	0	2	0	2	0
	1	0	<=3	0	<=3	0	0
		1	<=1	1	<=1	1	0
	2	0	<=2	0	<=2	0	0
		1	0	1	0	1	0
7	0	0	<=5	0	<5	0	0
		1	<=3	1	<=3	1	0
		2	<=1	2	<=1	2	0
	1	0	<=4	0	<4	0	0
		1	<=2	1	<=2	1	0
		2	0	2	0	2	0
	2	0	<=3	0	<=3	0	0
		1	<=1	1	<=1	1	0
	3	0	<=2	0	<=2	0	0
		1	0	1	0	1	0

Table. 2: Comparison of messages exchanged between MFC, DLFM and FLINK

n	MFC				DLFM			FLINK	
	#dl	#m	#dn	#msg	#dl	#m	#msg	#m	#msg
20	9	0	9	400	18	0	724	9	760
	6	3	5	558	9	4	742	9	760
40	19	0	19	1600	38	0	3044	19	3120
	14	7	10	2312	19	9	3082	19	3120

60	29	0	29	3600	58	0	6964	29	7080
	22	10	15	5266	29	14	7022	29	7080
80	39	0	39	6400	78	0	12484	39	12640
	29	14	20	9422	39	19	12562	39	12640
100	49	0	49	10000	98	0	19604	49	19800
	37	18	25	14776	49	24	19702	49	19800

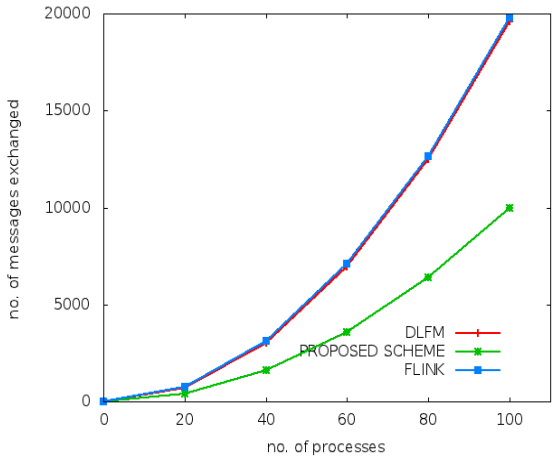

Fig. 5: Experimental Result I

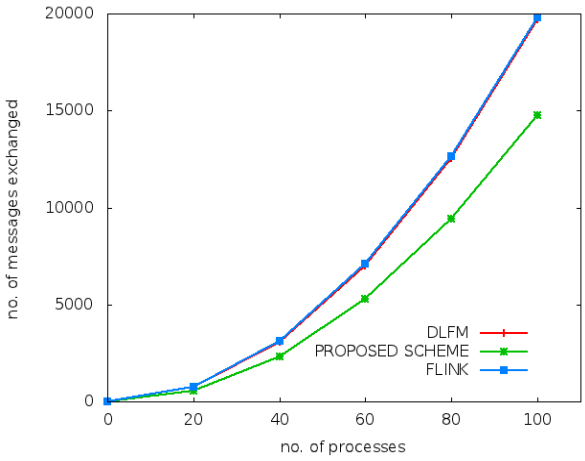

Fig. 6: Experimental Result II

7. References

[1] Lamport L., Pease M., Shostak R., Reaching Agreement in the presence of Faults, Journal of ACM, Vol.27, pp. 228- 234, Apr 1980.

[2] K.Q. Yan and Y.H. Chin, ”An Optimal Solution for Consensus Problem in an Unreliable Communication System,” *Proceedings of International Conference on Parallel Processing*, Aug. 1988, pp. 388-391.

[3] F.J. Meyer, and D.K. Pradhan, ”Consensus with Dual Failure Modes”, IEEE Transactions on Parallel and Distributed Systems, vol. 2, no. 2, April 1991.

[4] M. Dalui, B Chakraborty, B. K. Sikdar, “Quick Consensus Through Early Disposal of Faulty Processes”, *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics San Antonio*, TX, USA - October 2009.

[5] S.C. Wang and K. Q. Yan, “Revisit Consensus Problem on Dual Link Failure Modes”, *Proceeding of Computer Software and Applications Conference*, 1998. COMPSAC ’98, pp. 84 - 89, 19-21 Aug 1998, Vienna, Austria.

[6] D. Dolev, “Unanimity in an Unknown and Unreliable Environment,” IEEE FOCS, 1981.

[7] M. Fischer, ”The Consensus Problem in Unreliable Distributed System (A Brief Survey)” Lecture Notes in Computer Science, *Proceeding of the 1983 International FCTConference*, Borgholm, Sweden, pp. 127-140, Aug. 1983.