Volume 6, Issue 4, April 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Agreement Protocol with Dual Failure Problem

Monalisa Dey*, Sainik Kumar Mahata
Computer Science and Engineering Department, JIS College of Engineering,
West Bengal, India

Abstract— This work reports an efficient solution of reaching agreement in a fully connected distributed system. The proposed approach considers both the communication link and node failures. Detection of the faulty links and nodes make this scheme more efficient. Faulty nodes are disposed early thus leading to a quicker solution. The introduced network partitioning scheme further improves the performance by drastically reducing the message exchange overhead. Results establish that the proposed solution not only significantly reduces the message complexity, but also enhances the fault- coverage capability of a system

Keywords— Consensus, Early Disposal, Malicious and Dormant Link Faults, Malicious and Dormant Node Faults, Network partitioning.

I. 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. However reaching agreement becomes more complicated whenever some of the components in the system are faulty. The faulty components try to confuse the non-faulty components by sending arbitrary value or by not sending any value at all. 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 called the Byzantine Agreement [1]. Another related problem, the consensus problem has also been studied extensively in [2], [3]. 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 the initial value of the source node. 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], or link failure [2], [5]. However none of them have considered both node and link failures simultaneously. This encouraged us to propose a solution MFC (maximum fault coverage) to the consensus problem assuming both link and node failure. The symptoms of a faulty link and node can be classified into two types: dormant or crash and malicious (arbitrary).

Yan and Chin [2] had proposed a solution to the consensus problem by protocol FLINK, but they had treated all link failures as malicious. S. C Wang and K. Q Yan [5] had solved this problem by protocol DLFM but they had dealt with only dormant and malicious link failures. Also these protocols are unable to detect any of the faulty components in the network. However in this paper we have considered the unreliability of both the communication links and the nodes and we have devised a method for the detection of the faulty components. MFC can tolerate t node faults with $n > 2t$, where n is the total number of nodes in the distributed system. It 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 $O(n^2)$.

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. In our scheme we can not only reach consensus with both node and link failures, malicious and dormant in the system, we can also detect the faulty nodes and links. Further, a network partitioning scheme is introduced to handle large networks.

II. THE PROPOSED MODEL

In a large distributed network reaching a common decision among the nodes require huge information exchange. If the faulty nodes can be identified early the whole process can be speeded up. A link may be damaged and the message sent by it may be changed or delayed. The proposed model takes care of both dormant and malicious faults in nodes and links, and the system can reach an agreement in just two rounds of message exchanges.

In the first round, all nodes exchange their own initial values and at the start of the second round every node pi knows the initial value of all the n-1 nodes. Each node i then arranges these values in a vector vi. That is, Vector for node i is vi=[v1,v2,..,vi,..,vn]. In the second round each node sends their vector vi to all other n-1 nodes, a 2d array MATi is constructed at each node i. MATi is constructed using vector vj as the j-th column in the array. In a system having faulty nodes and links the arrays created are not the same. A node, by inspecting the array can identify the faulty components.

Any node detects all the faulty node(s) by discarding the corresponding rows and columns from its array. 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 then it decides on majority of 0's or 1's in the row thereby forming majority matrix MAJ. Say for node i majority value of the k-th row after eliminating entries other than 0 or 1 is MAJk. By inspecting their individual majority matrices 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. A partitioning scheme is introduced with the target to further reduce message exchanges while reaching an agreement. It results in low congestion and high network throughput. The introduction of partitioning in this process is described in Section IV.

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

1) Fd and Ld are the number of dormant faulty nodes and links respectively.
2) Fm and Lm 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 = Fd + Fm 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:

- Array MAT_i of size nxn and array v_i of size n stored at each
- dec_i= decision taken by each node.

First round: Each node sends its initial value to all. If a node i don't 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.

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 MATi [j][x] ! = M then
9: decrement c.
10: if MAT_i [k][j] = = Ai[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] != vk then
24: if MAT_i [j][k] != vj 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 Ai 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.

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 vi = 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 link57 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 from decision making. 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].

p1 = 0, p2 = 0, p3 = 1, p4 = 1, p5 = 0, p6 = 1, p7 = 1

Fig. 1 Initial Values

P1	P2	P3	P4	P5	P6	P7
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

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.

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.

MAT_1=
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

MAT_2=
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

MAT_3=
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

MAT_4=
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

MAT_5=
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

MAT_6=
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

MAT_7=
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

Fig. 3 2d arrays formed after second round

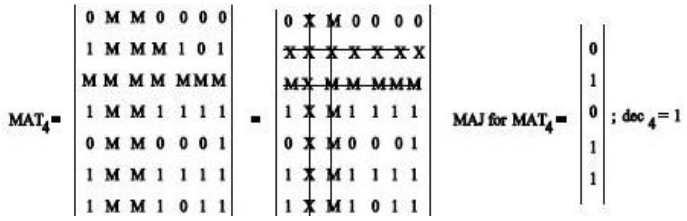

Fig. 4 Decision Making

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 vi of node i will be broadcasted to the other nodes. There are at most $\lceil ((n - t) - d - 3) / 2 \rceil$ malicious faulty 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 n-t-(d+1) non-M's and at least $\lfloor (n - d + 1) / 2 \rfloor$ (greater than $\lceil ((n - t) - (d + 1) + 1) / 2 \rceil = \lceil ((n - t) - d) / 2 \rceil$ 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 vi of node i will be broadcasted to the other nodes. There are at most $\lceil ((n - t) - d - 3) / 2 \rceil - 1$ malicious faulty links and d dormant links in the system as if $d \geq 1$ and n is in even. After the second round, node j receives at least (t + d + 1) M's and at least $(n - (t + d + 1)) - (\lceil ((n - t) - d - 3) / 2 \rceil - 1) = \lfloor ((n - t) - d + 1) / 2 \rfloor + 1$ 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 $\lfloor ((n - t) - d + 1)/2 \rfloor + 1$ (greater than $\lceil (n - (t + d + 1) + 1)/2 \rceil = \lfloor ((n - t) - d)/2 \rfloor$ the majority required when n is in even) vi's in the i-th row, so, the majority value in the i-th row should be equal to vi.

Theorem 1: The decision taken by each node i is 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.

By the method of early disposal, disposing off the rows and columns corresponding to the dormant and malicious faulty nodes are done. These faulty nodes do not take part in constructing MAJ_i. Each row k in array Ai is nothing but the values that node k sends to each node i. Since the total number of malicious faulty links in an n node system containing d dormant link faults can be maximum $\lceil ((n - t) - d - 3) / 2 \rceil$, the maximum number of faulty values in each row cannot be more than $\lceil ((n - t) - d - 3) / 2 \rceil$. Thus while calculating the majority value of each row k, the correct initial value of each node k is always obtained. This proves that the MAJ_i computed has the correct initial values of all nodes, and hence the decision taken by inspecting it is always correct.

III. 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 linkij is malicious faulty and node i is malicious faulty. Node i sends correct values to all other nodes k (k=1 to n-2) 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, so it would send correct value to all except node i. Any other node k while inspecting row i in it is 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 it is 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 $m \leq \lceil ((n - t) - d - 3) / 2 \rceil$, 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 faulty 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

IV. IMPROVEMENT THROUGH PARTITIONING

A distributed network of n nodes is partitioned into a number of groups, say g. A node of a group G_i is selected as the leader and initiates the process of consensus (Algorithm 1) within G_i. The rounds of message exchanges to reach a local agreement among the members of G_i is called local round. Once the local rounds for all the groups are completed, one node (leader) from each group G_i then participates, considering the weighted local decision value Xi, in a process to reach the global agreement. The weighted decision value is represented as X_i= (d,w), where d (decision value), 0, 1 and w = number of non-faulty nodes in the group G_i. At the initialization phase of network partitioning, a randomly selected node say, Q_r (initiator broadcasts an initialization message. After receiving it, each node of the system initializes a counter to 1 and starts incrementing. The node G_i then further broadcasts g tokens. A node with a counter value greater than 0, accepts a token and then resets its counter to 0. This ensures that each token can be received by one and only one node. The nodes holding the tokens are the leaders. Each leader logically forms a group of (n / g) - 1 nodes.

Algorithm 2:

Input: Leaders of the groups and their weighted local decision-value.

Output: Global-decision-value.

1) If for any k , 1 < k < g (g is the number of groups), local round for group Gk is finished leader Q of G_k sets a random timer Q[T] and starts decrementing it.
2) If Q doesn't receive any advertisement from an initiator of global round and Q[T] = 0 then Q initiates the global round by sending local-decision and weight (i.e. no. of non-faulty nodes in Gk) to all the leaders.
3) If a leader R ∈ G_i receives initialization message from Q then R resets R[T]=0 and participates in the global round by exchanging its weighed local decision value.
4) Q initiates Algorithm 1 considering the set of leaders as a group and decision value computed is the global decision-value.
5) The leader L of each group conveys global-decision value to all processes belonging to its group Gl.
6) Return global-decision-value.

In global round (Algorithm 2), if a leader of a group G is faulty, there may be a chance of mishap. However, the proposed scheme can mask off such faults as the faulty leader is detected in the local round, a new leader is selected from G following a cellular automata based election algorithm reported in [6]. The global round is then restarted with the newly elected coordinator. If all the nodes in a group is faulty then that group will not take part in the global agreement at all as described in [7]. If the non-faulty leader notes that the condition $n > 2t$ is not satisfied and it already receives a global round initiation message, it computes decision-value = majority of the initial values of non-faulty nodes and sends this decision value weighed by the number of non-faulty nodes in the group.

V. PERFORMANCE STUDY

This section reports performance evaluation of our proposed scheme.

The number of message exchanges required in the MFC (Algorithm 1 & 2) is

$$m = 2n^2 / g + 2g^2 + (n - g - f) \quad (1)$$

The first and the second terms represent the number of messages. The 3rd term (n − g − f) appears due to the fact that after completion of global round, the leaders of each group informs the global decision value to all the non-faulty processes belonging to that group. Therefore

$$\frac{dm}{dg} = \frac{-2n^2}{g^2} + 4g - 1$$

And,

$$\frac{d^2m}{dg^2} = \frac{4n^2}{g^3} + 4 > 0$$ (g and n are both positive)

For optimum m,

$$\frac{dm}{dg} = \frac{-2n^2}{g^2} - 1$$

i.e. $g^3 = \frac{g^2}{4} + \frac{n^2}{2}$ i.e $g = \frac{1}{4} + \frac{n^2}{2g^2}$

i.e. $g \approx \sqrt[3]{\frac{1}{2}n^2}$ (2)

VI. EXPERIMENTAL RESULTS

The fault tolerance and coverage capability of MFC is compared with DLFM and FLINK in Table I. The first column represents the total number of participating nodes n. Column 2 (no. of tolerable node faults), column 3 (tolerable malicious faulty links) and column 4 (tolerable dormant faulty links) show the results of MFC. Column 5 (tolerable malicious faulty links) and column 6 (tolerable dormant faulty links) that DLFM can tolerate. Column 7 (tolerable malicious faulty links) and column 8 (tolerable dormant faulty links) provides the same FLINK can tolerate.

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 in Table II. The first column represents the number of participating nodes (n). Two sets of observations for total number of dormant nodes and dormant links are taken. Columns 2 (no. of dormant links), column 3 (no. of malicious links), column 4 (no. of dormant nodes) and column 5 (messages exchanged) give the result of MFC. Column 6 (total 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. The comparison between MFC with partitioning and DLFM and FLINK is shown in Table III. The first column represents the number of participating nodes, column 2 (no. of groups), column 3 (no. of tolerable faulty nodes) and column 4 (messages exchanged) gives the result of MFC with partitioning. Column 5 (messages exchanged) give the

results of DLFM and column 6 (messages exchanged) show the results for FLINK. The results shown in the Tables I and II and III point to the fact that the proposed solution maximizes the fault coverage capability of a system. It is seen that as the size of the network and the number of dormant faults increase, the total number of messages exchanged are reduced and efficiency of MFC increases (Figure 5). With the introduction of the network partitioning scheme the message exchange overhead is reduced considerably (Figure 6). The impact of number of partitions is illustrated in Figure 7. It clearly shows that the number of message exchanges reduces with the increase in number of partitions. This carries on up to a threshold limit.

Table I Comparison of the Fault coverage Capability between MFC, DLFM and FLINK

n	MFC			DLFM		FLINK	
	t	m	d	m	d	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 III Comparison of No. 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

Table III Comparison between MFC with Partitioning DLFM and FLINK In Terms of Message Exchange Complexity

n	MFC			DLFM	FLINK
	#groups	#fp	#msg	#msg	#msg
20	3	8	294	760	760
	4	9	239	760	760
	5	7	218	760	760
25	3	11	446	1200	1200
	4	10	356	1200	1200
	5	12	308	1200	1200
35	3	17	850	2380	2380
	4	15	660	2380	2380

	5	17	553	2380	2380
	3	19	1103	3120	3120
	5	17	708	3120	3120
40	10	14	536	3120	3120

After that it cannot reduce message exchange anymore. This can be explained from the analytical results. From the expression of m (Equation 1), we can observe that, for g << n; the 1st term prevails and the contribution of 2nd term is negligible. If we go on increasing the g, contribution of the 1st term decreases but the 2nd term becomes more prominent and the 3rd term decreases. So, there must exist a minimum in the curve for m (since d2m/dg2 > 0). Graph shown in Figure 7 displays that for 40 processes if the number of groups is 9, the message exchanges is minimum (531). The expression for g, shown in Equation 2, also conforms to that value.

That is, g ≃ ≃ 9.2 ≃ 9 3 2 40 21

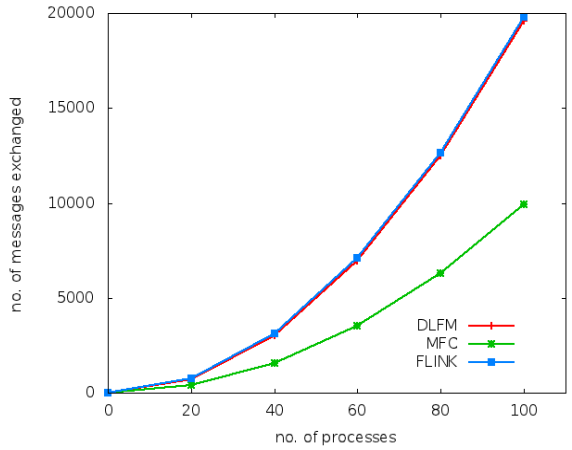

Fig. 5 Experimental Result 1

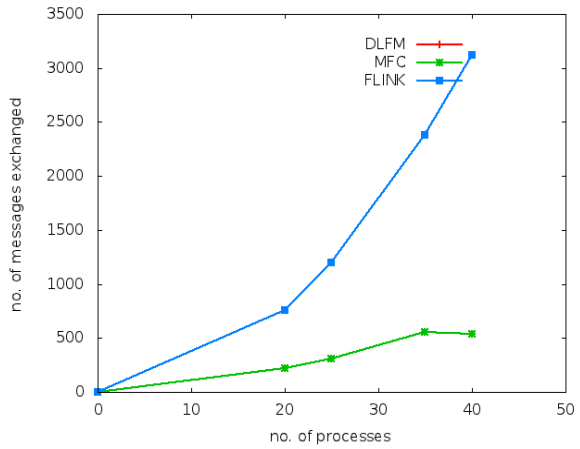

Fig. 6 Experimental Result 2

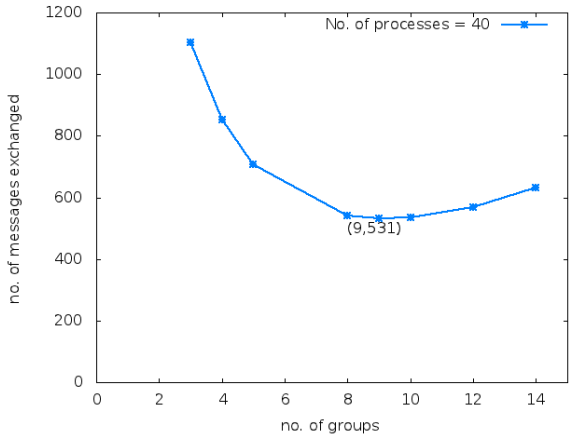

Fig. 7 Experimental Result 3

VII. CONCLUSIONS

In a distributed system, in reality, both the nodes and links of network can be subject to a fault. These faults can again be categorized into dormant and malicious faults. Our proposed scheme 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. A correct decision is ensured with just two rounds of message exchanges. It gives a better fault coverage compared to 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 transparent. Moreover we can see that the number of messages exchanged to reach consensus is also lesser than the previous solutions and as the number of dormant faults increases, the efficiency of our scheme increases. Further a network partitioning scheme is also introduced for further reduction in message exchange overhead. Results establish that the proposed scheme can drastically reduce the message complexity in comparison to the previous solutions.

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] Kalyan Mahata, Meghnath Saha, and Sukanta Das, "Cellular Automata Based Coordinator Selection Scheme in Distributed System " accepted in International Conference on Scientific Computing (CSC'09).

[7] Mamata Dalui, Suhrid Bakshi and Biplab K. Sikdar, "CA based quick consensus in distributed system Through network partitioning", Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Istanbul, Turkey, 10-13 October 2010.

