Simple and Faster Consensus Approach in Fault Tolerance

Yu Sanda Win, and Ei Ei Hlaing

***Abstract*—**The distributed agreement algorithms require to reach the goal that all the non-faulty processes have to achieve consensus on some issue and to ascertain that consensus within a finite number of steps. All the members of group in most consensus agreement algorithms have to process with the coordinator to reach the agreement. To doing so, these members have to send the messages via matrix in each site. The proposed system uses simple vector values to detect the faulty processes in a minimum number of message exchanges. Moreover, the system can save the time of counting vector for each matrix. Therefore, the system can quickly detect the faulty processes that are removed before the final agreement. Network partitioning is used to reach the final agreement in a large network.

***Keywords*—**Byzantine General Problem, Consensus Problem, Distributed Computing, Fault Tolerance, Network Partitioning.

I. Introduction

IN the Distributed Computing, message exchanging among the processors to get the system reliability is essential. The important point in message exchange is making agreement among the non faulty processes. In a system, the faulty processors and failed components may send wrong message to other part. Therefore, agreement and consensus problems are very important in distributed message passing system. The final goal of these agreement problems is to have all non faulty processes reach the same value of agreement. The reach this point, all the processor in the system must be participated.

There are many consensus agreement problems for reaching an agreement in a distributed system. Among them, the famous one is Byzantine General Problem [4]. In this problem, each processor in the group has to send the message to each other. It may include the traitors that send different and wrong message to different other processes and some components may fail in arbitrary way.

Due to the various natures of the processors, the system may become more difficult. There are many solutions that are tried to get the final agreement with the maximum fault tolerance and minimum message exchange. But, some delays may occur because of message exchange overhead [8]. The proposed system can tolerate the maximum faults in the system and can quickly detect the faulty process with the minimum message exchange by using the vector value to reach agreement. Moreover, the proposed system can reduce the half of counting time to become vector count.

Yu Sanda Win is a Ph. D candidate from the University of Technology (Yatanarpon Cyber City), Pyin Oo, Lwin, Myanmar.email:yusandawin@gmail.com

Ei Ei Hlaing is from the Computer University (Taung Ngu), Taung Ngu, Myanmar.

II. Related Works

The conventional Byzantine agreement problem [1] with n processors requires many numbers of rounds to exchange the messages. It has the upper bound of n>=3t+1 where t is the maximum number of tolerable faulty processes. It is proved that all the conventional byzantine agreement problems require t+1 rounds for sending message among the processors.

Synchronous consensus with Mortal Byzantines approach [5] reduces the number of rounds for sending messages up to n>=2t. However, it has an assumption that faulty process takes a finite number of arbitrary steps before they eventually crash.

Quick consensus algorithm [7] gets the agreement only in two rounds of message exchange. It uses the network partitioning scheme to reduce the message exchange overhead and network traffic. Each process exchanges the initial value in first rounds. In the second round, the processes get the vector of all others to calculate the decision vector. But, that algorithm can handle the faulty processes which happened only in the first round. However, that algorithm did not consider for the faulty processes in the second round of message exchange.

The next solution for consensus problem proposed in [8] also has only two rounds of message exchange. It can solve the assumption proposed in Quick consensus algorithm. The algorithm can detect the faulty processes in the second round. But that has some delay to take the agreement value.

The faster consensus agreement approach [9] solved the problem in simple and better way to detect the faulty processes in two rounds of message exchanges. This system can get the maximum fault tolerance and minimum message exchange. But, the processors have to exchange messages via matrix to get the final decision value. It can be effect the system performance.

The vector based simple consensus approach [10] proposed the faster algorithm to reach the final agreement with the maximum fault tolerance. This algorithm used only simple vector to calculate the agreement value. But, this algorithm has the time delay counting the vector value to become vector count 0 and 1.

The proposed system can mainly solve the delay of counting vector values. It can reduce the time to count vector to form the matrix of each site up to half time. It can quickly reach the final agreement by comparing the matrix and vector count 0.

III. Proposed Algorithm

The main point of consensus problem is that all the faulty processes have to reach the same decision value within the maximum fault tolerance and minimum message exchange in a large network. The proposed system provides the consensus agreement only in two rounds. All the processors do not need to compare with the matrix of coordinator. It is only needed to compare the vector values for reaching the agreement. Moreover, the system can detect the faulty processes and can reach the consensus within the short time via vector values and only one vector count.

A. Making Consensus for a Processor Group

This algorithm is used to detect the faulty processes in a group of processor and to make a decision among the processors within that group. This can detect the faulty processes only in two rounds of message exchange. All the processors must have their initial values (0/1) to deicide the final agreement. The sample case assume that, a system includes five processors (P_1, P_2, P_3, ..., P_5) in which P_1 and P_5 are assumed faulty.

Each processor exchanges its initial value to each other in first round. After exchanging the initial value in round one, in second round, each processor Pi possesses the initial value of every other and then the initial vector F_i is composed by these values. The vector F_i stored at each process site.

Each process site has to receive all vectors from other processes to inform the two dimension matrix Si.

Each matrix has to count the number of 0 in every row value to get the vector count for comparing the coordinator matrix. By comparing this vector count and coordinator matrix, the first decision is obtained quickly. This first decision does not save for any case. Therefore, this decision has to compare the coordinator matrix to become the final decision. And then, final result vector is obtained with the consensus values of non faulty processes and the information of the traitors.

There are four cases depending on the various natures of the faulty processes. These cases are:

Case 1:In the system, each process receives the initial value of others correctly if no one processor in the system is faulty as shown in Fig.1. In this case, all the rows values in the matrix are identical. Therefore, the decision value can be computed easily.

Case 2:In case 2, the faulty process sends conflict value in round one. It is assumed that such case is not happened in round two. All the matrices for each process site are shown in Fig.3(a). In these matrices, the row values of faulty processes are not identical with others.

Case 3:In this case, the faulty processes send wrong values to the other processes only in round 2. All processes received the correct values in round one. This case is shown in Fig.4.

Case 4: The processors in this case have to receive wrong values of faulty processes in both rounds. After finishing first round, the row values of faulty processes are not identical with the rows values of others. This case is Fig.5(a). After both rounds had finished, the row values and 0,1 vectors of each Si are shown in Fig.5(b).

The next section is described the solution to reach a consensus among the non-faulty processes. To reach this consensus approach, the following assumptions are needed.

(i) All the processors in the network are fully connected. The entire links among the processors are reliable and any delay does not be introduced.

(ii) Assume two types of fault can be included. First is dormant faulty process 'd' that may be caused from traitors. The other is crash faulty process 'c' that cannot send back any message when a process site crashes permanently.

(iii) t = d + c is the total number of faulty processes.

Algorithm I:

P_{co} = Coordinator processor
P_i = Process ID at each processor site, i=1,2,...,n. where n= total number of processor
F_i = Initial Vector for each site after first round
S_i = n x n Matrix for each site after second round
$V0_i$ = Vector for total number of message'0' for processor i
D = Decision Vector
A = Agreement Result Vector
Input : Process ID P_i and initial value (0/1) for each process
Output : Decision and Agreement Result

Step 1:
- P_{co} = randomize (P_i)
- Each process sends its initial value to all other processes
- If a process P_i does not receive message from P_{co}, Then P_{co}=randomize(Pi) cv=i where $P_{co} \notin \{P_i\}$
- If a process P_i does not receive message from any other process P_k
  then $F_i[k] = C$

Step 2:
- Each process sends its vector F_i to every other processor
- If process P_i does not receive any vector from any other process P_k
  then
  for i= 1, 2, .., n of F_i
  { S[k][i]=C}

Step 3:
For each processor with P_{co}
D_i = TRUE;
For j=1 to n
Begin
If ($V0_{co}[j] != V0_i[j]$)
Then D_i=FALSE and Break;
End if
End
Step 4:
For P_{co} only
For i=1 ton n
If D_i=TRUE then

```
        Begin
            Init=S[i][i]
            For j=1 to n
                If Dj== TRUE AND Init!= S[j][i] then
                    Di=FALSE
             End if
            End
        End if
    End
```

Step 5:

```
    For j=1 to n
    Begin
        If Dj= TRUE, Then
          Aj=Sco [cv][j]
        Else Aj=Dj
    End
```

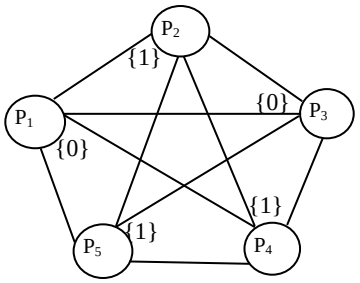


Fig. 1 All process sites their initial values

Case 1:

0	1	0	1	1
0	1	0	1	1
0	1	0	1	1
0	1	0	1	1
0	1	0	1	1

Fig. 2 No faulty process as all S_i are identical ($S_1=S_2=S_3=S_4=S_5$)

Case 2:

0	1	0	1	0	V0$_1$	3	3	1	2	2
1	1	0	1	0						
1	1	0	1	1						
0	1	0	1	1						
1	1	0	1	0		T	T	T	T	T

S_1 Decision Vector

Fig. 3(a) Matrix of each process site that is identical with wrong values of each faulty process

0	1	0	1	0
1	1	0	1	0
1	1	0	1	1
0	1	0	1	0
1	1	0	1	0

S_1

Decision Vector

F	T	T	T	F

Result Vector

F	1	0	1	F

Fig. 3(b) Final decision vector and result vector by comparing with the coordinator matrix

Case 3:

P1	0	1	0	1	1	F$_1$
P2	0	1	0	1	1	F$_2$
P3	0	1	0	1	1	F$_3$
P4	0	1	0	1	1	F$_4$
P5	0	1	0	1	1	F$_5$

Fig. 4 First round with no faulty process

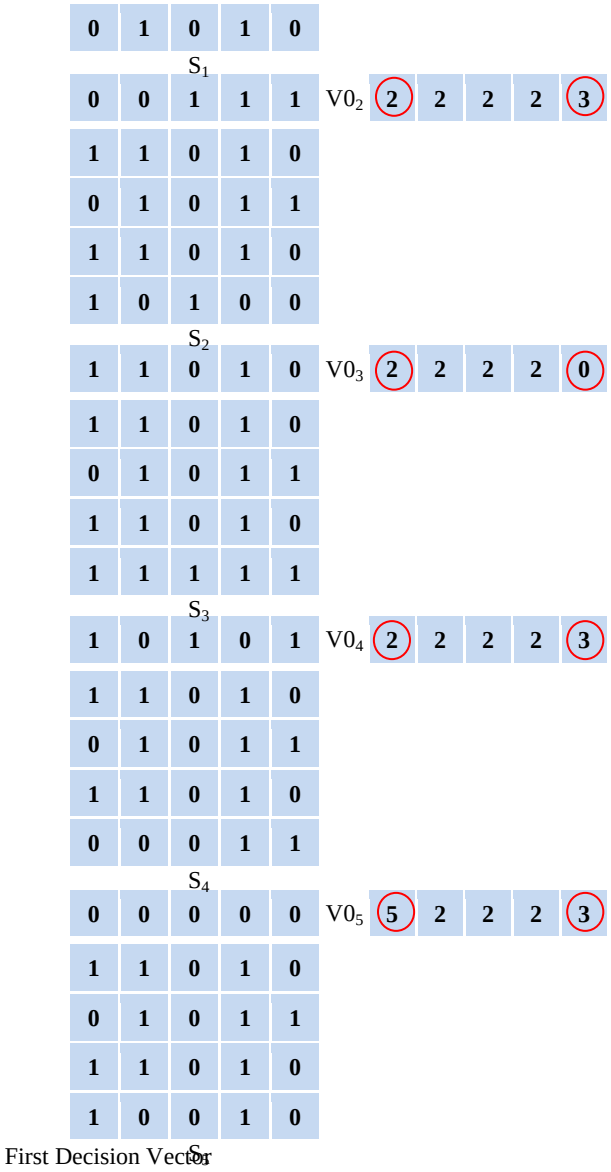
Each process site receives wrong vectors of faulty processes in second round and it is assumed that P1 and P5 are faulty processes. According to the steps, the decision vector and result vector are calculated as the same in case 4.

Case 4:

P1	0	1	0	1	1	F$_1$
P2	1	1	0	1	0	F$_2$
P3	0	1	0	1	1	F$_3$
P4	1	1	0	1	0	F$_4$
P5	0	1	0	1	1	F$_5$

Fig. 5(a Vector F_i of each process site after first round

1	0	1	1	0	V0$_1$	2	2	2	2	3
1	1	0	1	0						
0	1	0	1	1						
1	1	0	1	0						



First Decision Vector

F	T	T	T	F

Fig. 5(b) First decision vector that is calculated by using vector 0

0	0	1	1	1
1	1	0	1	0
0	1	0	1	1
1	1	0	1	1
1	0	1	0	0

Decision Vector

T	F	T	F	T

Result Vector

1	F	0	F	0

Fig. 5(c) Final result vector that is only in coordinator matrix

B. Making Consensus for Entire Network

It is sure that the network traffic is huge in a large network. Assume that G number of groups is partitioned in a large network and N processors are included in each group. Each group has a coordinator C_i to decide the consensus agreement among its members. After completing the above *algorithm I*, the result vector is decided for that particular group. Once the local round for all the groups are completed, the coordinator from each group participates in a global decision making round. After executing the above *algorithm I*, each processor possesses a value of decision vector F and number of non-faulty processes W for its group G_i.

At the initialization phase of network partitioning, a coordinator C_i for each group is selected randomly and they broadcast an initialization message [2][3]. After receiving it, each process of the system initializes a counter value to 1 and starts incrementing. Then C_i broadcasts G tokens. Each token can be received by one and only one process. The processors holding the token are leaders. Each leader logically forms a group of N/G-1 processes.

Algorithm II:

Input: Leaders of the groups and their weighted local decision value (F, W)

Output: Global decision value

(i) if for any k, 1<k<G, local round for Group G_k is finished. C_k sets random timer T[C] and starts decrementing it

(ii) If C doesn't receive any advertisement from an initiator of global round and T[C]=0, then C send local decision and weight to all leaders
Else go to step 2

(iii) If a leader Q ∈ G_i receives initialization message from C then Q resets T[Q]=0

(iv) C initiates *Algorithm I* considering the set of leaders as a group and decision value computed is the global decision value

(v) The leader of each group conveys global decision value to all processes belonging to its group G_i

(vi) Return global decision value

If weighted local decision value is sent by the faulty leader of group G, 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. Once the faulty leader is detected, a new leader is selected from G. The new leader participates in global round by sending a reply to the initiation message.

IV. Performance Evaluation

The following is the analytical results of our proposed algorithm in terms of number of message exchanges to reach consensus.

N = number of processors in a network
G = number of partitioned groups in a network
n = number of processors in each group that means n= $\frac{N}{G}$

T_f = total number of faulty processes in each group
m = number of messages
(i) Message exchange for local round,

$Tm_L = (n - 1)$

Tm_L represents the number of message exchange required to get for local consensus. In each group, there is assumed that n processors. Each processor i send its own message to all others and others sent back.

(ii) Message exchange for global round,

$Tm_G = (G-1)$ (1)

Tm_G represents the number of message exchange required to get global consensus.

(iii) Message exchange for coordinator,

$Tm_C = n - T_f$ (2)

Tm_C represents the number of message exchange required by the coordinator of each groups to inform the global decision to all members except the faulty processes belonging to that group.

For m = 1,

Total message exchange,

$$Tm_T = Tm_L + Tm_G + Tm_C \quad (3)$$
$$= (n - 1) + (G-1) + n - T_f$$
$$= n + G - T_f$$

For m = q,

Total message exchange,

$$Tm_T = n + G - T_f + q \quad (4)$$

The different number of processors N is partitioned into different number of Groups G. In our experiment, we assumed that the partitioned group is started from 3 and initialized number of processors in a network is 40.

The performance of the proposed scheme following figure is compared with the faster consensus agreement [9] in terms of message exchange overhead while reaching an agreement. In faster consensus algorithm, all processors site have to compare with the coordinator via matrices of initial value. It can cause large amount of message exchanges. The proposed algorithm can overcome the sending of vast amount of messages. This system uses only the vectors to get the decision value. Using this vector approach can reduce the number of message exchange that shown in following figure.

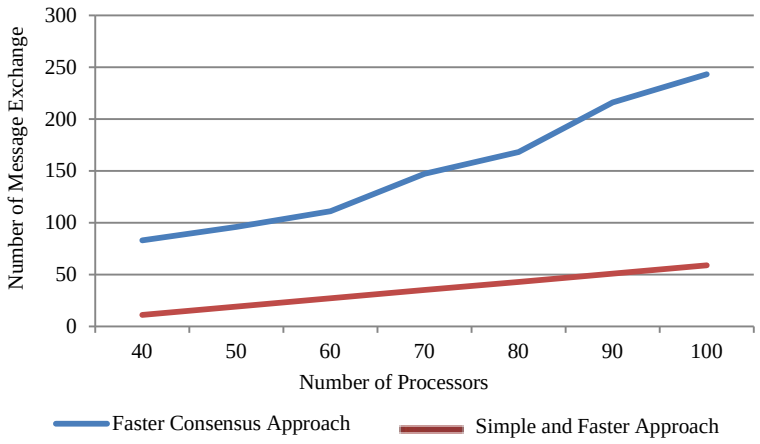


Fig. 6 Comparison between Faster Consensus and Simple and Faster Approach

V. Conclusion

The simple and faster consensus approach can give highly fault tolerance and can reduce the number of message exchange overhead. In this approach, the processors in the group do not have to exchange the message via matrix to reach the final agreement value. Moreover, it can reach the final agreement only in two round of message exchanges in a fully connected network while the total number of faulty processors is less than or equal to (n-1)/2. (Where n is the total number of faulty processors in a given network). The proposed system can detect the faulty processors by using simple vector of each processor and using only one vector count for fast decision. In a short time of visit, the final decision value can be detected within a very few number of message exchanges. For large networks, network partitioning scheme is used for message exchange overhead.

Acknowledgment

The author would like to special thanks to her supervisor, her teachers, her family and all her friends for their encourage and special advice for this paper.

References

[1] Lamport L.,Pease M.,Reaching Agreement in Presence of Faults', Journal of ACM,Vol.27 pp.228-234,Apr,1980.
http://dx.doi.org/10.1145/322186.322188

[2] Raymond K.,'A Tree Based Algorithm for Distributed Mutual Exclusion', ACM Trans.Computer Systems,1989.

[3] M.Singhal, N.G. Shivaratri 'Advanced concepts in operating systems: distributed, database, and multiprocessor operating systems', Tata McGraw- Hill,2001 Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[4] Andrew S.Tanenbaum and Maarten van Steen, Distributed System Principles and Paradigms, International Edition, 2002.

[5] J.Widder,G.Gridling,B.Weiss,'Synchronous Consensus with Mortal Byzantine' in Proceedings of the 37th Annual IEEE/IFP International Conference on Dependable Systems and Networks.

[6] Kalyan Mahata, Meghnath Saha, and Sukanta Das, 'Cellular Automata Based Coordinator Selection Scheme in Distributed System', accepted in CSC'09, USA, 2009.

[7] M. Dalui, B Chakraborty, B. K. Sikdar, 'Quick Consensus through early disposal of faulty processes', Submitted in Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics San Antonio,TX, USA - October 2009.

[8] Anirban Roy, Nishad T M, Sangeetha. K. G, R.Muthamil Selvi, Arpan Modal, 'Fault Tolerant Apporach for Reaching Consensus in a Distributed System', 2011.

[9] Yu Sanda Win, Ei Ei Hlaing, 'Faster Consensus Agreement Approach for Fault Tolerance in a Distributed System', International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 2, February 2013)