Byzantine Fault Tolerance In The Distributed Environment Using Markov Chain Technique

R. Kalaivani

ABSTRACT: The abstract of this paper is to tolerate the byzantine fault by providing the predefined constraints of the Nodes in the distributed environment. The nodes in the distributed environment automatically generated their constraints using Markov chain. The distributed environment predefined constraints and the member nodes predefined constraints can be updated periodically. According to this update, if the member nodes predefined constraints may not matches with the distributed system predefined constraints then using Breadth First Search technique the membership service discards the service of the node in the distributed environment . The new node having constraints wants to communicate with the distributed environment. These constraints can be compared with the distributed system constraints using probability of random matching technique.

1 INTRODUCTION

Byzantine fault is an arbitrary failure occurs in the distributed environment causes heavy damage to the system. The word “Byzantine” refers to Byzantine generals problem which can be used in army in the past decades[1].Byzantine problem can be tolerated only there are 3N+1 nodes in the distributed environment[15]. For example, Dynamo uses tens of thousands of servers located in many data centers around the world to build a storage back-end for Amazon’s S3 storage service and its e-commerce platform[11,13].Byzantine fault can be tolerated using distributed Byzantine Quorum systems techniques to provide security in the database storage[2,14].This Quorum system can be integrated with protocol for proactive recovery of servers[3,18].But it cannot be completely eradicated and also understand. By using some techniques it can be tolerated but it is not sufficient in the recent days. Byzantine fault tolerant algorithms will be increasingly important in the future because malicious attacks and software errors are increasingly common and cause faulty Nodes to exhibit arbitrary behavior[4,19]. In distributed system byzantine failure is used to describe the worst possible failure semantics in which any type of error occur[5,17]. The distributed environment uses membership service. The membership service is used it periodically notifies of other system nodes of membership changes[6,16].The membership service is used dynamically. It allows only authorized service to communicate with this group of system in the distributed environment[12].To reduce this byzantine fault occurred in the distributed environment provide the predefined constraints in the distributed system. The membership service allows only the service with possible constraints related to the distributed system constraints.

- *R.Kalaivani, M.E.Second year, Department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tirunelveli. Tamilnadu. Email: kalaivani202@gmail.com*

The predefined constraints can be automatically generated by the system using Markov chain[7]. The probable constraints can be analyzed with the distributed system using random matching technique[8]. There are many set of constraints in the system in distributed environment. These constraints can be easily identified using searching techniques.

2 DESIGNING THE ARCHITECTURE:

The Byzantine fault can be tolerated by using the techniques of this paper. There should be more than three nodes in the distributed environment. It is impossible to tolerate the problem if 3N+1 nodes is not in the distributed environment. There are set of predefined constraints is used in the distributed environment. The nodes wants to communicate with the member node of distributed system or the node wants to join the distributed environment. The particular node also has the predefined constraints. The membership service checks that constraint is related to the distributed system constraints. This comparison can be done by random matching technique. The membership service searches the constraints in the distributed system using searching technique. The predefined constraints can be generated using Markov chain.

3 UTILITIES OF MEMBERSHIP SERVICE

The membership service is used in the large scale distributed environment. The membership service groups the nodes in the distributed system. This membership service checks periodically all the Node in the distributed environment. The Node occur byzantine fault it can be removed by the membership service. The node in the distributed environment has a predefined set of constraints. If any new node wants to communicate with the distributed system, the membership service checks the new node constraints if it matches with the distributed environment constraints then allow the node to communicate. The predefined constraints in the distributed system can be generated automatically by the Node or human can give instructions manually.

4 PROBABLE CONSTRAINTS

There are n numbered predefined constraints in the nodes in distributed system is put into random order n! arrangements have equal probabilities. The number of matches in the random variable Sn has the values 0,1,2,….. By using random matching technique[8], compare the new node constraints and the distributed system constraints in

the distributed environment. The probability of having m matches is given by

P[0]=1-1+1/2!-1/3!+…….+/-1/(N-2)!+/-1/(N-1)+/-1/N!
P[1]=1-1+1/2!-1/3!+……+/-1/(N-2)!+/-1/(N-1)!
.
P[N-2]=1/(N-2)!{1-1+1/2!}
P[N-1]=1/(N-1)!{1-1}=0
P[N]=1/N!

In this probability there are m correct matches and N-m incorrect matches can be arranged. The probability of exactly m correct matches is given by

$$bm=\binom{N}{m}\frac{(N-1)^{N-m}}{N^N}$$

The probability of random matching technique is used to perform the probable constraints.

5 SEARCHING TECHNIQUES:

There are several nodes in the distributed environment. These nodes can be previously checked by the membership service it matches the predefined constraints in this environment and then join these nodes in the distributed environment. These predefined constraints can be changed periodically. Each node in the distributed environment has also predefined constraints that constraints also updated periodically. Suppose the newly generated predefined constraints of the node in the distributed environment cannot matches with the predefined constraints in the distributed environment. The membership service discards the node in the distributed system. For this purpose the membership services uses breadth first search(BFS) technique[10]. Choose any node in the distributed environment designate it as a search node. Determine the unvisited adjacent node for the search node. The membership service checks all the adjacent node's predefined constraints. In this way the process continues until all the nodes in the distributed environment can be checked by the membership service. The node predefined constraints cannot matches with the predefined constraints in the distributed environment that node can be rejected by the membership service.

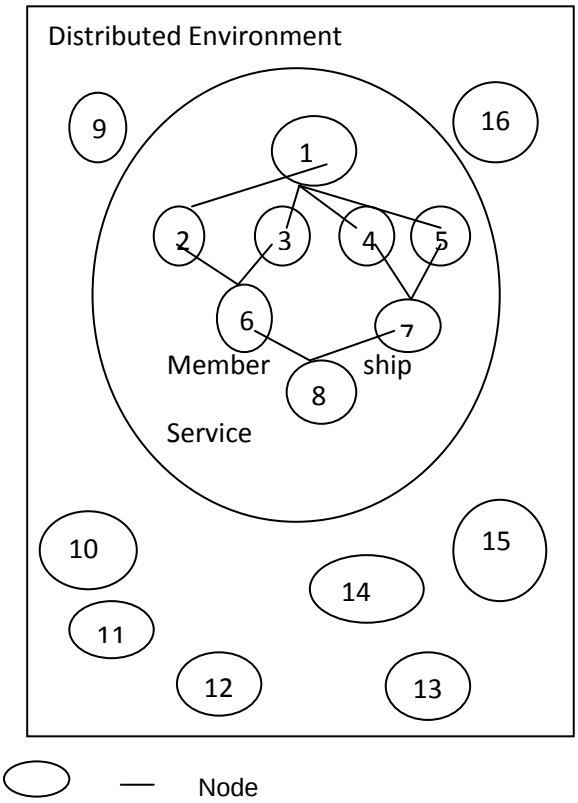


Fig.1 *Membership Service searches the node using BFS.*

The features of the algorithm are the membership service first checks there are more than three nodes in the distributed environment, if it does not have three nodes byzantine problem cannot be solved. So it cannot continue the process and terminates it. Otherwisethe membership service searches all the nodes in the distributed environment using breadth first search technique. If any node constraints cannot matches with the distributed environment predefined constraints that node can be deleted. The insertion and deletion of nodes in the distributed environment can be performed using this algorithm. The algorithm for searching the node in the distributed environment using BFS technique by the membership service is given below:

1. Initialize
   Visited[]=0;
   Predefinedconstraints pc;
   Adjacent node v;
   Node n;
   Location of the node loc;
2. if(N>=3n+1) then
   continue the following steps
   else goto step 13
3. BFS traversal on the node is carried out beginning at N;
4. Mark N as visited
   Visited[N]=1;
5. Check visited node satisfies predefinedconstraints
6. If(Visited[N]!=pc)
   {
7. Position a pointer q on (loc-1) node
   i.e q=firstnode;
   for(i=1;i<loc-1;i++)

```
    q=q->nextnode;
8.  Delete the desired node
9.  Stop
    } Else
10. find adjacent nodes for the beginning node N
11. if(!Visited[v])
    Mark v as visited
    Visited[v]=1;
12. process terminate until all nodes are visited
13. stop the process
```

This is the algorithm for adding the new node in the distributed environment by the membership service that is given below:

1. Acquire memory for new node with its address in pointer p
2. Assign value to the node.
3. Position a pointer q on (loc-1) node
   i.e. q=firstnode;
   for(i=1;1<(loc-1);i++)
   q=q->nextnode;
4. Insert the node after the pointed by q
5. Stop the process.

6 NODE GENERATED CONSTRAINTS

The node generated constraints can be periodically changes. The new fault can be tolerated by the recently updated predefined constraints. The predefined constraints generated depends on previously generated constraints existed for the last two days and not on past predefined constraints[9]. This technique can be used by Markov chain. The recently developed constraints is in the state i, there is a fixed probability Pij that will be future constraints in state j.

$$P\{Xn+1=j/Xn=i,Xn-1=in-1\ldots\ldots,X1=i1,X0=i0\}=Pij$$

For all states i0,i1,……..in-1,i,j and all n>=0.Such a process is known as Markov chain[7]. Let p denote the matrix of one step transition probability Pij so that

$$P=\begin{pmatrix} P00\ P01\ P02\ldots \\ P10\ P11\ P12\ldots. \\ \ldots\ \ \ldots \\ \ldots\ \ \ldots \\ Pi0\ Pi1\ Pi2\ldots \end{pmatrix}$$

If the constraints generated recently, it can be used in the past α and if the constraints generated recently, it cannot be used in the past β. The one step transition probability is given by

$$\begin{pmatrix} P= \alpha & 1-\alpha \\ \beta & 1-\beta \end{pmatrix}$$

The one step transition probability is defined. The n-step transition probability is given by

$$Pij=P\{Xn+m=j/Xm=i\},n>=0,i,j>=0\}$$

The chapman-kolmogorov equation provides a method for computing these n-step transition probabilities.These equations are

$$Pij^{(n+m)}=\sum Pik^n\ Pkj^m \text{ for all } n,m>=0 \text{ for all } i,j$$

Let p(n) denote the matrix of n-step transition probabilities Pij^n then

$$P^{(n+m)}=P^n\ P^m$$

The predefined constraints can be updated periodically by using Markov chain.

7 SIMULATION

Let us assume that the probability of newly generated constraints used the past prefined constraints, α=0.7 and the newly generated constraints cannot be used the past constraints, β=0.4, to find the probability that the newly generated constraints can be used the past 16 days constraints.

$$P=\begin{pmatrix} .7 & .3 \\ .4 & .6 \end{pmatrix}$$

Probability of generated predefined constraints for past two days

$$P^2=\begin{pmatrix} .7 & .3 \\ .4 & .6 \end{pmatrix}\begin{pmatrix} .7 & .3 \\ .4 & .6 \end{pmatrix}$$

$$=\begin{pmatrix} .61 & .39 \\ .52 & .48 \end{pmatrix}$$

P^2=0.6100

Probability of generated predefined constraints for past four days

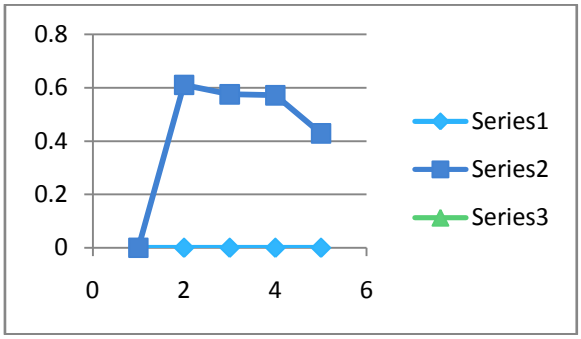
$$P^4=\begin{pmatrix} .61 & .39 \\ .52 & .48 \end{pmatrix}\begin{pmatrix} .61 & .39 \\ .52 & .48 \end{pmatrix}$$

P^4=0.5749

The tabulated results for the probability of generating predefined constraints using past constraints.

Probability of generated predefined constraints for past days	P^2	P^4	P^4	P^16
Probability obtained	0.6100	0.5749	0.5714	0.4286

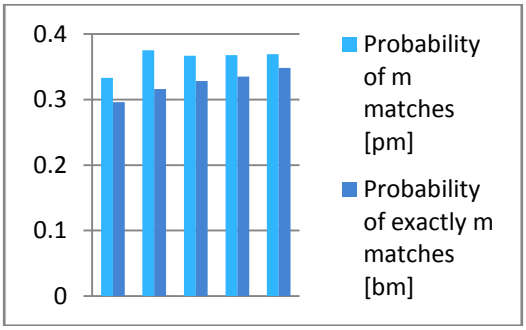
The graphical representation of the probability of generating predefined constraints using the past constraints is



Using the predefined constraints the membership service decides which node can be inserted or deleted. This can be performed using the algorithm is given in the section 4. The random matching technique is used to find the probability of m correct matches in the constraints in distributed system. The tabulated results are presented below.

	N=3	N=4	N=5	N=6	N=10
Probability of m matches [pm]	0.333	0.375	0.367	0.368	0.369
Probability of exactly m matches [bm]	0.296	0.316	0.328	0.335	0.348

The graphical representation for the probability of correct matches in the distributed environment is



8 CONCLUSION AND FUTUREWORK

The byzantine fault can be tolerated by providing the predefined constraints in the distributed environment. The constraints can be automatically generated by the node is performed by using Markov chain. By providing the searching techniques the constraints can be searched in the nodes of the distributed system using membership service. The new node constraints and the nodes of the distributed system constraints can be easily compared and the probable constraints can be easily identified by probability of random matching technique. The future work of this paper is that the new node without create fault in the distributed environment is requested to join in the environment but this constraints is not matched in the nodes of the distributed system constraints that node is rejected. So the future work is to create the new correct constraints in the distributed environment automatically.

REFERENCES

[1] L.Lamport,R.Shostak and M.Pease. The Byzantine Generals Problem.ACM Transactions on Programming Languages and Systems,4(3),1982.

[2] R.Kalaivani. Byzantine Fault Tolerance using distributed Quorum Systems. Proceedings of National Conference on Recent Trends in Advanced Computing in Manonmaniam SundaranarUniversity,Tamilnadu,India,2013.

[3] 3.Eduardo A.P.Alchieri,Alysson Neves Bessani,Fernado Carlos Pereira and Jonida Silva Fraga. Proactive Byzantine Quorum Systems. Brazil University of Lisban.

[4] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,”Proc. Third Symp. Operating Systems Design and Implementation(OSDI ’99), Feb. 1999.

[5] George Couloris,Jean Dollimore and Tim Kinderberg. Distributed Systems, Fourth Edition, pg.no.51,461-466.

[6] Barbara Liskov and Moses Liskov. Automatic Reconfiguration for Large Scale Reliable Storage Systems. IEEE Transactions on dependable and secure computing, VOL. 9, NO. 2, MARCH/APRIL 2012.

[7] Sheldon M.Ross. Introduction to Probability Models, Fourth Edition, pg.no.135-140.

[8] Feller. An Introduction to Probability Theory and its Applications, Third Edition,VOL.1, pg.no.107-109.

[9] Dr.A.Singaravelu and Dr.S.Sivasubramanian. Probability and Queueing Theory, pg.no.3.36-3.39.

[10] P.Revathy,S.Poonkuzhali. Data Structures, Pg.No.7.10-7.13.

[11] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A.Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W.Vogels, “Dynamo: Amazon’s Highly Available Key-Value Store,”Proc. 21st ACM Symp. Operating Systems Principles, pp. 205-220,2007.

[12] J. Dean,, “Designs, Lessons and Advice from Building LargeDistributed Systems,”Proc.Third ACM SIGOPS Int’l WorkshopLarge Scale Distributed Systems and Middleware (LADIS ’09), Keynotetalk, 2009.

[13] Amazon S3 Availability Event, http://status.aws.amazon.com/s3-20080720.html, July 2008.

[14] K. Birman and T. Joseph, “Exploiting Virtual Synchrony inDistributed Systems,” Proc. 11th ACM Symp. Operating SystemsPrinciples, pp. 123-138, Nov. 1987.

[15] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H.Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Servicefor Internet Applications,” Proc. ACM SIGCOMM, 2001.

[16] M. Reiter, “A Secure Group Membership Protocol,” IEEE Trans.Software Eng., vol. 22, no. 1, pp. 31-42, Jan. 1996.

[17] H.D. Johansen, A. Allavena, and R. van Renesse, “Fireflies:Scalable Support for Intrusion-Tolerant Network Overlays,” Proc.European Conf. Computer Systems (EuroSys ’06) , pp. 3-13, 2006.

[18] J. Cowling, D.R.K. Ports, B. Liskov, R.A. Popa, and A. Gaikwad,“Census: Location-Aware Membership Management for Large-Scale Distributed Systems,” Proc. Ann. Technical Conf. (USENIX’09), June 2009.

[19] D. Oppenheimer, A. Ganapathi, and D.A. Patterson, “Why DoInternet Services Fail, and What Can Be Done About It?” Proc.Fourth USENIX Symp. Internet Technologies and Systems (USITS ’03),Mar. 2003.