Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Distributed Fault Tolerance System in Real Time Environments

Lakshmi Prasad Saikia
Professor and Head
Deptt. of Computer Science and Engg
Assam down town University, India

Kundal Kr. Medhi
Master of Engineering
Deptt. of Computer Science and Engg.
Girijananda Choudhury Inst. of Mgmt and Tech., India

Abstract: Fault Tolerance is an important issue in Distributed Computing. Fault-tolerant describes a computer system or component designed so that, in the event that a component fails, a backup component or procedure can immediately take its place with no loss of service. A fault can occur for many reasons like communication failure, resource or hardware failure or it may be of sometime for process faults which occur due to shortage of storage of resource, software bugs etc. This eventually leads to a fault environment. In real time distributed system feasibility of task is very important because there is a deadline defined for each task and should be finished on or before its deadline even there is a fault in the system. This paper aims to provide a better understanding of fault, fault tolerance and fault tolerance techniques used in the distributed real time environments.

Keywords: Fault Tolerance, Real Time Distributed System, Fault Environment, faulty system, fault-free system.

I. INTRODUCION

We can define Distributed System as a collection of independent computers that appear to the users as a single computer. Its main goal is to present a single system image. It can be easily expandable by adding new computers which are hidden to users. Distributed system can be effectively defined as a collection of multiple autonomous computers that interact with each other over a communication channel in order to achieve their desired goal or result. Common characteristics of a Distributed System are Resource Sharing, Openness, Scalability, Transparency, and most importantly is Fault Tolerance. In distributed system the individual workstations communicate each other by passing messages.

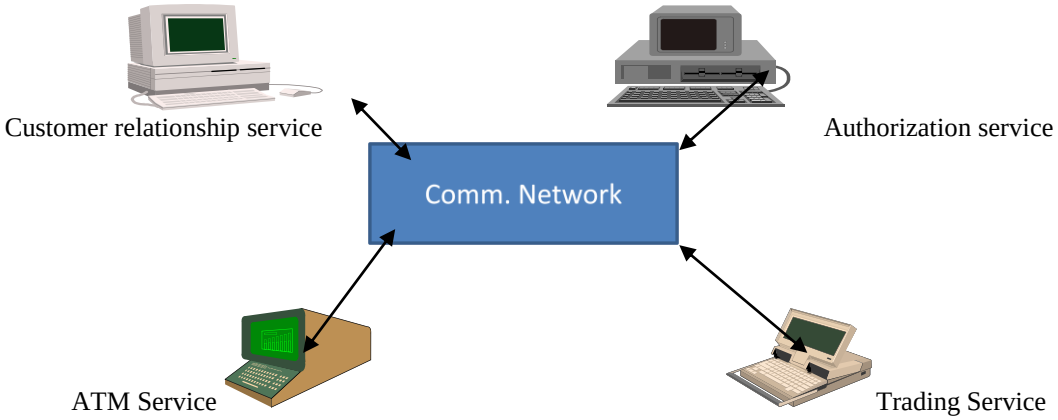

Fig 1: A Distributed System

Fig 1 shows the working environment of a distributed system. A Real time system can be defines as a system that is highly dependable on deadlines. Deadlines in the sense that a time will be given to particular task and the task should be completed with that time period. So, in order to function a real time system should be reliable and a task running on real time system should be feasible, reliable, and scalable. Some examples of real time system are Nuclear Systems, Robotic Controls, medical equipments, defense systems etc. so, it is now clear to us that a real time system not only depends upon the logical outputs but also on the time of production of a output. Aircraft control is a good example of a real time distributed system. So, a distributed system can be effective for feature like that it can be present like a single system image; it can be easily expandable when load to a system occurs. And more importantly failure of one component can be covered by another one, so as to reduce fault can give a better output. This property called Fault Tolerance, which we will discuss in next Section III.

II. FAULT AND FAULT ENVIRONMENT

In this section we will discuss about some basic about faults and fault environment and finally fault tolerance. Before moving up to fault tolerance, first let us review to some basic concepts of faults and fault environment.

In order to give a better performance and to give a logical output, a system must detect the faults and perform even in case of faults. There are different types of faults which can occur in a real time distributed system. These faults can be broadly classified as: Network faults, Physical fault, media faults, process faults. Network faults occur in a network due to network partition, packet loss, communication failure etc. Physical faults can occur in hardware like fault in CPUs, memory fault etc. Media faults occur due to media head crashes. Process faults occur due to shortage of resources, software bugs etc. But, Fault occurs with respect to time are as follows:

Permanent: These failures occur by accidently cutting a wire, power breakdown and so on which can cause major disruptions and some part of system may not be functioning as desired.

Intermittent: These failures appear occasionally. Mostly they are ignored while testing the system and only appear when the system goes into operation.

Transient: They are caused by some inherent fault in the system. However, these failures are corrected by retrying roll back the system to previous state such as restarting software or resending a message. These failures are common to in computer systems. But, in real time system, main focus is on hardware fault tolerance.

Due to presence of faults, the system encounters many problems during execution or processing of any event. This ultimately leads to a ***Fault environment.***

III. FAULT TOLERANCE

Fault Tolerance can be defined as a property of a system which provides the facility to perform efficiently even in case of any faults. Fault tolerance can be achieved by detecting a faulty process, saving and restoring the computational tasks of the faulty processor, and then distributing the recovered task to the remaining processors so that the system can continue to operate, although with degradation of computing power. An appropriate fault detector can avoid loss due to any link failure, resource failure or in any other fault environment. Hardware Fault tolerance can be achieved by adding extra hardware like processors, resources like memory, I/O devices etc.

For tolerating any fault, first we require to detect the fault occurred in the system and then isolating it to the appropriate unit as quickly as possible. The main detection mechanisms are: Sanity monitoring, Watchdog monitoring, Protocol Fault, Transient Leaky Bucket counters. If a unit is really faulty, many fault triggers will be generated for that unit.

A. NEED OF FAULT TOLERANCE

The needs of fault tolerance are mentioned below:

- Better outcome of results in case of any faults.
- For reliable processing of transaction.
- To avoid faulty systems.
- Limit ourselves to types of failures and errors which are more likely to occur.

IV. RELATED WORK

Research and experimentation efforts began in earnest in the 1970s and continued through 1990s, with focused interest peaking in the late 1980s. A number of distributed operating system were introduced during these period; however, very few of these implementations achieved even modest commercial success. Different authors have reviewed the concept of fault tolerance computing system, like Ramamoorthy [1967], Short [1968], Avizienes [1971], Khul and Reddy [1980, 1981], Bagchi and Hakimi [1991]. The SAPO Computer built in Prague, Czechoslovakia was probably the first Fault-Tolerant Computer built in 1950-1954 under the supervision of Antonin Svoboda, using relays and a magnetic drum and was operated in 1957-1960.

Leslie Lamport [1974], has described the usual method of obtaining fault tolerance synchronization in distributed system is to add timeouts to time independent algorithms. A process sets a timer whenever it begins waiting for another process and a failure is assumed to have occurred if a certain period of time elapses without a response from the other process. According to the assumption, a distributed system as a network of processes joined by communication links is modeled. Each process executes an event driven algorithm, where an event is the arrival of a message or the process's clock reaching a certain value. The use of timeouts rests upon the following assumption.

Assumption: For any event *e* that causes process *i* to send a message to process *j*, there is a δ such that if the event e occurs at time *T* and a process *i* and *j* and the communication link joining them are non-faulty, then the message arrives are either both measured according process *i*'s clock, or are both measured according to process *j's* clock.

According to J. Laprie [1985], the service provided by a system is the system behavior as it is perceived by another system interacting with considered system.

Bagchi and Hakimi [1991], presented an algorithm for diagnosing faulty processors in arbitrary networks. Initially, each fault free processor knows only about itself and its physical neighbors. Fault-free processors start the algorithm by walking up and initiating the formation of a tree based testing topology. Multiple trees that are being formed simultaneously are merged into one. Diagnosis information is sent along with the new message that forms the tree. The authors show the algorithm is optimal in that it requires at most 3n log (p) + O (n+pt) message, where is the number of fault-free processors that start the tree and is the number of faulty processor. Christian [1993], said that before discussing fault tolerance, we need to design the building blocks and the failure these blocks would experience. So, the design of a fault tolerant system would be easier and effective in any fault environment.

In 1997, Andre Schiper said that a system failure occurs when the system behavior is not consistent with its specification. According to him, "more the number of components more are the things that could be faulty."

Felix Gartner [1999], mentioned redundancy as a key to Fault Tolerance. According to him: no matter how well designed or how fault tolerant a system is, there is always the possibility of failure if the faults are too frequent or too severe. The central observation here is that to be able to tolerate faults, one must employ a form of redundancy. He also gave another definition that a distributed system *A* is said to be redundant in space iff for all executions *e* of *A* in which no faults occur, the set of all configurations of *A* contains configurations that are not reached in *e*. M. Castro and Barbara Liskov [1999], presented a new replication algorithm that is used to tolerate Byzantine Faults. This offers both liveness and safety provided at most [(n-1)/3] out of the total of n-replicas are simultaneously faulty. This means that Clients eventually receive replies to their request and these replies are correct according to linearizability. The algorithm works in synchronous system like the Internet and it incorporates important optimization that enables it to perform efficiently.

In 2000, W.T. Pomales, gave an example of Fault Tolerant Architecture: A Primary Flight control computer. Here, the computers are connected to the flight control data buses that serve to exchange information among ***fly-by-wire*** system components. The Fly-by-wire system components enable the creation of artificial airplane flight characteristics that allow crew workload alleviation and flight safety enhancement.

In 2001, Richard Golding and Elizabeth Borowsky, presented a paper which describes one component of a self-healing storage system: the component that allows for automatic recovery of access to data when the power comes back on after a large-scale outage. The failure recovery protocol is a part of a suite of modular protocols that make up the Palladio Distributed storage System. This protocol guarantees that service will be repaired quickly and automatically when enough failures are repaired. The Pelladio divides the implementation into four protocols: *Access protocols, layout retrievalprotocol,reconciliation protocol, layout control protocol.*

In 2003, Yevgeniy Gershteyn, presented that Fault tolerance can be improved by replicating agents. Replication of service leads to increase level of complication of the system. The article examines "the use of transparent agent replication, and technique in which the replicates of agents appear and act as one entity thus avoiding an increase in system complexity and minimizing additional system loads."

In 2004, Paul Ezhilchevan presented that Failure Detectors are important building blocks for constructing fault-tolerant distributed system. Presence of failure detection can help us in recognizing the exact fault and in its diagnosis too.

P.M. Melliar Smith and L.E. Moser 2004] discuss progress in the field of real time fault in different way by considering synchronous and asynchronous fault tolerant designs, maintaining replica consistency, alternate fault tolerant strategies, including checkpoint restorations.

Lee Pike and J .Maddalon [2004] presented four kinds of abstraction for design and analyze of fault tolerant distributed system. These four abstractions concern system messages, faults, fault-masking, voting and communication. They are: message abstraction, fault abstraction, masking abstraction, and communication abstraction. Message Abstraction addresses the correctness of individual messages sent and received. Fault Abstraction addresses the kinds of faults possible as well as their effects in the system. Fault masking abstraction address the kinds of local computation processes make to mask faults. Communication abstraction address the kinds of data communicated and the proportion required for communication to succeed in the presence of faults.

In 2007, L. P. Saikia and K. Hemachandran, discussed the problem of distributed diagnosis in arbitrary network failures and repairs. The idea behind incorporating a fault tolerance capability to a distributed system is to provide the system with extra (redundant) resources. The purpose of this study was to simulate a distributed system and carry out fault diagnosis under Arbitrary Network topologies. In this algorithm, nodes detect failure in neighboring nodes and then propagate this information to other nodes in the network in two discrete steps: detection and dissemination. The failure information the nodes propagate consists of failure events, where a failure event is defined to be a transition of a node from *fault-free to faulty or faulty to fault-free.*

In 2008, A. Persya and T.R. Gopalakrishnan Nair showed an algorithm. It increases utilization speed and efficiency of scheduling. Deadline scheduling means that the task with the earliest required response time is processed. The most common scheduling algorithm are: Rate Monotonic (RM) and Earliest Deadline First (EDF). This paper deals with the interaction between the fault tolerant strategy and the EDF real time Scheduling Strategy.

Sumin Park and Kwangyong Lee [2010] designed a Fault Tolerant System based on Runtime Behavior tracing. It was implemented on Linux 2.6.24 kernel. Its main objective is to improve the reliability of operating systems and have been focusing on the evolution of the kernel architecture or protecting against device driver errors. The research as experimented on GP2X-WIZ mobile game player and experimented to solve a frame buffer problem when playing a game emulator.

Arvind Kumar et al [2011], presented a paper investigating the different techniques of fault tolerance. The main focus is on types of faults occurring in the system, fault detection and recovery techniques. A system can behave after failure in three (3) such ways: (i)Fail Stop System, (ii)Byzantine System and (iii)Fail-Fast System. They have mentioned some approaches for fault tolerance in Real Time distributed system. They are:

a) Replication
    - Job Replication
    - Component Replication
    - Data Replication
b) Check pointing
c) Scheduling/Redundancy
    - Space Scheduling/Redundancy

- Time Scheduling/ Redundancy.
- Hybrid Redundancy.

In 2012, Sourabh Dave and A. Raghuvanshi in their paper provides a study of fault tolerance techniques in distributed system, especially replication and checkpointing. Replication in simple definition means to make several copies of that dataand keep them on several nodes. Checkpointing is primarily used to avoid losing all the useful processing done before a fault has occurred. Checkpointing consists of intermittently saving the state of a program in a reliable storage medium. According to them, a system failure occurs when the system behavior is not consistent with its specification. A system consist of several components, more the number of components, more are the things that could be faulty. Since failures are caused by faults, a direct approach to improve the reliability of the system is to try to prevent fault from occurring into a system. This approach is called fault prevention. The other approach is the fault tolerance. The goal is to provide service despite the presence of faults in the system.

In 2012, Ban M. Khammas designed a fault tolerance for soft real time distributed system (FTRTDS). This system is designed to be independent on specific mechanism and facilities of the underlying real time distributed system. The research was done in a special environment chosen to implement and fast to work. It was programmed in Visual C++ language. The computers were connected by a Ethernet network with 100Mbps speed and the topology is star connection. The FTRTDS has a distributed unit and a central unit. The Distributed Fault Tolerance Unit (DFTU) was distributed on all computers in the system and it is run with the system boot of that computer. The Central Fault Tolerance Unit (CFTU) which is run manually by user in one computer chosen for. It needs backup software for each part of distributed system to use it when it is necessary.

In 2012, Sanjay Bansal et al, presented a paper for simulation framework for evaluation of fault tolerant large dynamic distributed system. They have proposed a Saturn, a multithreaded process oriented over simulation framework which is designed for modeling large scale distributed system. Realistic simulation is provided by it to provide a wide range of distributed system technologies. The solution is based on several proposed extension to the simulation model of the MONARC (Models of Networked Analysis at Regional Centers) simulation framework. This extension refers to fault tolerance and system orchestration mechanism in order to access the reliability and availability of distributed system. It is a simulator which also evaluates major QoS of the heartbeat based adaptive failure detection mechanism.

In 2013, Pankaj Saxena and Kapil Govil designed an optimized algorithm for enhancement of performance of Distributed Computing system. In Distributed Computing system, the whole workload is divided into small and independent units, called tasks and it allocates onto the available processors. In this paper a simple algorithm for task allocation in terms of optimum time or optimum cost or optimum reliability is presented where the number of tasks are more than the number of processors. For obtaining the optimal time or cost or reliability for each task initially the emphasis will be on those modules of tasks which have the maximum probability of data transfer. In case of time and cost, the elements will be added and in case of reliability they will be multiplied.

H. Kanemitsu and Masaki Hanada [2013] presented a paper proposing a method for effective use of computational resources each of which has a single processor in a heterogeneous distributed system. The method automatically derives set of mapping between each processors and each assignment unit (i.e., the set of tasks in a Directed Acyclic Graph program such as a workflow type job), by which the degree of contribution for each processor towards the response time minimization is maximized.

Lakshmi Prasad Saikia et al [2013] presented a paper giving a detailed literature survey in Fault Tolerance for Distributed Computing Systems and thereby discussing about the faults, error and failure that a distributed system can face.

V. CONCLUSION

Fault tolerance assesses the ability of a system to respond gracefully to an unexpected hardware or software failure. It is very difficult to detect fault in distributed system as compared with uniprocessors. Fault Tolerance techniques are also depending upon its occurrence. There are three things in real time distributed system which are to be kept in mind when fault tolerance is applying. These are reliable, scalable and feasible. In real time distributed system feasibility of task is very important because there is a deadline defined for each task and should be finished on or before its deadline even there is a fault in the system.

REFERENCES

[1]. Ramamoorthy C.V., "*A Structural theory of machine diagnosis*", Spring Joint Computer conference Proc. Montvale, NJ, pp.743-756, 1957.
[2]. Short R.A., "*The Attainment of reliable digital system through the use of redundancy-A survey*", IEEE Computer Group news, pp. 2-17, March, 1968.
[3]. Avizienis A., "*Fault Tolerance Computing-An overview*", IEEE Computer, Vol.4,pp. 5-8, Jan/Feb 1971.
[4]. Brian Randell, "*System Structure for Software fault tolerance"*, "IEEE Transaction on Software Engineering", "Vol.SE-1 No2", June 1975.
[5]. Khul J.G. and Reddy S,M., "*Fault- diagnosis in fully distributed system*" in Proc. 11th International Symp. Fault Tolerant Computing , pp 100-105, June 1981.
[6]. Leslie Lamport, "*Using Time instead of Time out for Fault tolerance distributed system*", "ACM Transaction on Programming language and system", "Vol 6 No2", April 1984.
[7]. Christian Flaviu, "*Understanding Fault Tolerant Distributed System*", May 1993.

[8]. T. Chandra and S. Toueg, "*Unreliable failure detectors for reliable distributed system*", "Journal of ACM", March 1996.
[9]. Felix C. Gartner, "*Fundamental of Fault Tolerance distributed Computing in Asynchronous Environment*", "Vol31 No 1", "ACM Computing Surveys", 1999.
[10]. Richard Golding and Elizabeth Borowsky, "*Fault Tolerant Management in large scale distributed storage system*", 1999.
[11]. Wilfredo Torres-Pameles, "*Software Fault Tolerance: A Tutorial*", "Langley Research Center, Virginea", October 2000.
12]. Yevgeniy Gershteyn, "*Fault Tolerance in Distributed System*", "Rochester Institute of Technology", Feb 2003.
13]. Paul Ezhilchevan, "*On the progress in Fault-Tolerant Real Time Computing*", Proceedings of the 23rd IEEE International Symposium on Reliable Distributed System 2004.
[14]. Lee Pike and J.Maddalon, "*Abstraction for fault Tolerance distributed system verification*", In Theorem Proving in Higher Order Logic(TPHOLs), vol3222 of lecture notes in Computer Science, Pages-257-270, Springer,2004.
[15]. Lakshmi Prasad Saikia and K. Hemachandran, "*Simulation of System Level Diagnosis in Distributed Arbitrary Network*", "Journal of Theoritical and Applied Information Technology", 2007.
[16]. Dona Fisman and Orna Kupferman, "*On verfying fault tolerance in Distributed Programming*", "Hibrew University, Jerusalem, Israel", 2008.
[17]. A. Persya and T.R.Gopalakrishnan Nair, "*Fault Tolerant Real Time System*", International Conference of Managing Next Generation Softwares Apllication (MNGSA-08), Coimbatore 2008.
[18]. Andrew S. Tananbaum, "*Distributed Operating System*", Prentice-Hall, 2010.
[19]. Sumin Park and K. Lee, "*Design of fault tolerant system based on Runtime behavior tracing* " University of Science and Technology, Korea, Feb 2010.
[20]. Arvind Kumar, Rama Shankar Yadav, Ranvijay and Anjali Jain "*Fault Tolerance in Real Time Distributed System*", "Vol3 No2", "International Journal on Computer Science and Engineering", Feb 2011.
[21]. Sanjay Bansal, Sanjeev Sharma, and Ishita Trivedi, "*A detailed review of fault tolerance technique in distributed system*", International Journal on Internet and Distributed Computing System, Vol1, No1, 2011.
[22]. R.A.Mamon and Y.Ryu,"*Building Scalable Fault tolerant network*", Myongi University, Feb 2012.
[23]. Sourav Dave and A. Raghuvanshi, "*Fault Tolerance Techniques in Distributed System*", "Vol 1, Issue 2", "International Journal of Engineering Innovation & Research", 2012.
[24]. Ban M. Khammas, "*Design of Fault Tolerance in Real Time Distributed System*", "Vol.8, No.1, PP11-17", "Al-Khwarizmi Engineering Journal", 2012.
[25]. Z. Xie, H.Sun,and K.Saluja, "*A Survey of Software fault tolerance Technique*.", University of University of Wisconsin-Madison, 2012.
[26]. Sanjay Bansal, Sanjeev Sharma, Ishita Trivedi, Ankita Verma, and Mirnalika Ghosh,"*Simulation of Framework of evaluation of fault tolerant large dynamic distributed system*", Council of Innovative Research , "International Journal of Computer and Distributed system", Vol1, Issue 2, Aug 2012.
[27]. Pankaj Saxena and Kapil Govil, "*An Optimised Algorithm for Enhancement of performance of Distributed computing system*", "International Journal for Computer Application", "Vol 64, No. 2", Feb 2013.
[28]. H.Kanemitsu and Masaki Hanada, "*Effective use of computational Resources in Multicore Distributed System*", "ICACT Transactions on Advanced Communications Technology (TACT)", "Vol 2, Issue5", Sept 2013.
[29]. Lakhmi P. Saikia , Nilotpal Baruah, and K.Hemachandran, "*System Diagnosis and Fault Tolerance for Distributed Computing system*", "International Journal for Computer Science and Communication Networks" Vol.3, Issue 5, 284-295, Oct 2013.

