Fault Tolerance in Real Time Distributed System

Arvind Kumar, Rama Shankar Yadav, Ranvijay, Anjali Jain

Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology, Allahabad

Abstract- **In this paper we investigate the different techniques of fault tolerance which are used in many real time distributed systems. The main focus is on types of fault occurring in the system, fault detection techniques and the recovery techniques used. A fault can occur due to link failure, resource failure or by any other reason is to be tolerated for working the system smoothly and accurately. These faults can be detected and recovered by many techniques used accordingly. An appropriate fault detector can avoid loss due to system crash and reliable fault tolerance technique can save from system failure. This paper provides how these methods are applied to detect and tolerate faults from various Real Time Distributed Systems.**

Key Words**: Fault Detection, Fault Tolerance, Real Time Distributed System.**

I. INTRODUCTION

A faulty system due to any reason during processing some task can causes some damages. A task running on real time distributed system should be feasible, reliable and scalable. The real time distributed system such as nuclear systems, robotics, air traffic control systems, grid etc. are highly dependable on deadline. A fault in real time distributed system can result a system into failure if not properly detected and recovered at time. These systems must function with high availability even under hardware and software faults. Fault-tolerance is the important technique used to maintain dependability in these systems. Hardware and software redundancy are well-known effective methods. Hardware fault-tolerance achieved through applying extra hardware like processors, communication links, resource (memory, I/O device) whereas in software fault tolerance tasks, messages are added into the system to deal with faults. Fault should be detected by applying reliable fault detector followed by some recovery technique. Many fault detection techniques are available but it is necessary to apply appropriate fault detector. Unreliable fault detector can make mistake by erroneously suspecting correct process or trusting crashed process. Rest of the paper is organized as: Types of fault are discussed in section II, Issues are explained in next section III and then the system behavior is explained in section IV. The approaches used to failure detection and fault tolerance are explained in section V and section VI respectively. Section VII concludes the paper followed by future work in next section.

II. TYPES OF FAULT

There are different types of fault which can occur in Real-Time Distributed System. These faults can be classified on several factors such as:
Network fault: A Fault occur in a network due to network partition, Packet Loss, Packet corruption, destination failure, link failure, etc.
Physical faults: This Fault can occur in hardware like fault in CPUs, Fault in memory, Fault in storage, etc.
Media faults: Fault occurs due to media head crashes.
Processor faults: fault occurs in processor due to operating system crashes, etc.
Process faults: A fault which occurs due to shortage of resource, software bugs, etc.
Service expiry fault: The service time of a resource may expire while application is using it. [1], [2], [3], [4], [5].

A fault can be categorized on the basis of computing resources and time. A failure occurs during computation on system resources can be classified as: omission failure, timing failure, response failure, and crash failure. Fault occurs with respect to time are shown in figure 1 as:

Permanent: These failures occur by accidentally cutting a wire, power breakdowns and so on. It is easy to reproduce these failures. These failures can cause major disruptions and some part of the system may not be functioning as desired.

Intermittent: These are the failures appears occasionally. Mostly these failures are ignored while testing the system and only appear when the system goes into operation. Therefore, it is hard to predict the extent of damage these failures can bring to the system.
Transient: These failures 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 very common in computer systems.

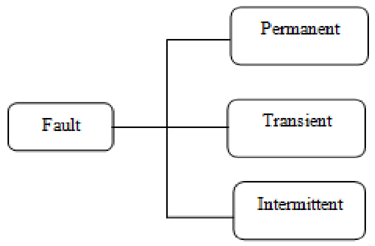

Figure.1: Types of fault in a system

Main focus is on hardware fault tolerance in real time distributed system. Software fault tolerance is often overlooked. This is really surprising because hardware components have much higher reliability than the software that runs over them. Most system designers go to great lengths to limit the impact of a hardware failure on system performance. However they pay little attention to the systems behavior when a software module fails. There are many different techniques for software fault tolerance (e.g. time out, audits, task rollback, exception handling, and voting). Most Real time systems must function with very high availability even under hardware fault conditions. The most useful hardware fault tolerance techniques are redundancy and load sharing. For tolerating any fault from the system 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 Faults, In-service Diagnostics, and Transient Leaky Bucket Counters. If a unit is really faulty, many fault triggers will be generated for that unit. The main objective of fault isolation is to correlate the fault triggers and identify the faulty unit.

III. ISSUES

In a distributed real time system or in general, fault tolerance [10] is the technique to give the required services in the presence of fault or error within the system. The aim is to avoid failures in the presence of faults and provide services as per requirement. In fault tolerance the fault is detected first and recovers them without participation of any external agents. The main issue in fault tolerance is how, where, and which technique is using to tolerate fault in distributed system. As we have seen many type of fault and failure arises in a system, so there should be an appropriate method which can tolerate such problem. In this paper will we will see various technique for tolerating different fault.
In any real time distributed system there are three main issues.

1. *Feasibility*- this means that a task running should be finished on its deadline even though there is a fault in the system. Dead line in real time system is the major issue because there is no meaning of such a task which is not finishing before its deadline. So the question is that which method is to be applied by which the task can finish on deadline in the presence of fault.
2. *Reliability*- in real time distributed system reliability means availability of end to end services and the ability to experience failures or systematic attacks, without impacting customers or operations.
3. *Scalability* –it is about the ability to handle growing amount of work, and the capability of a system to increase total throughput under an increased load when resources are added.

Now the question arise how these faults can be detected and removed or tolerated from different environment. A task running in distributed environment should be finished on its deadline. It may be hard or soft depend on task requirement. In hard deadline a task should be finished by its deadline sharply but in soft deadline task can finished nearby its deadline.

IV. FAILED SYSTEM BEHAVIOR

What will happen if a system failed? What are the affects of a system failure? Such behavior of a system has been explained in [13]. A system can behave after failure in three ways such as

- Fail Stop System
- Byzantine system
- Fail-fast system.

In fail stop system there is no output when a system fails. It immediately stops to sending any message or event and also does not respond any message receiving on network. Any failure in a system in fail stop manner can results a permanent fault in the system. Byzantine systems are those systems which not stop after failure but gives wrong output. These systems continue working without desired output. They may send wrong output or may respond later results transient type of fault. In Fail-fast system the system behaves like a Byzantine system for some time but moves into a fail-stop mode after a short period of time. It does not matter what type of fault or failure has caused this behavior but it is necessary that the system does not perform any operation once it has failed.

V. FAILURE DETECTION

Failure detection is the main issue in any system. Selecting a dependable failure detector is very difficult task. For detecting a fault accurately, a reliable fault detector is required. It can be removed by applying appropriate removing techniques. Reliability of fault detector and fault recovery method is depending upon the type of fault. For removing a fault/failure from the system it has to be detected first and then fault tolerance technique can be applied. Many failure detections have been explained in various papers as an independent service [19], [20], [21], [24], [25]. Many researchers have given fault detector for distributed system also. A failure detector has to provide good quality of service (QoS) but it is so far. Many schemes have been applied but none of them resolve it properly. A failure detection service must adapt to dynamic network conditions and application requirements. The failure detection service is implemented via variants of the Heartbeat mechanism [22]. The heartbeat mechanisms are as follows: Centralized, Virtual ring based, All-to-all, and Heartbeat groups.

VI. APPROACHES OF FAULT TOLERANCE

There are many approaches for fault tolerance in real time distributed system. A fault can be tolerated on the basis of its behavior or the way of occurrence. Following are the methods of fault tolerance in a system.

1. Replication
    a. Job Replication
    b. Component Replication
    c. Data Replication
2. Check-pointing
3. Scheduling/ Redundancy
    a. Space Scheduling/ Redundancy
    b. Time Scheduling/ Redundancy
    c. Hybrid Redundancy

A. Replication

Replication, is the process of sharing information by which it can ensure consistency between redundant resources (i.e. software or hardware components), to improve reliability, fault-tolerance, or accessibility. Job replication is the method of replicating job on multiple server such as in grid computing service is capable of receiving jobs, executing them, performing checksum operations on them, and sending the result back to the client [2], [14]. Figure 2 shows a distributed network in which every server S can communicate to each other and each server is connect to multiple clients C. A job can be distributed to servers for operation and result is back to the client. Data Replication is also commonly used by fault tolerance mech-anisms to enhance availability in Grid like environments where failures are more likely to occur.

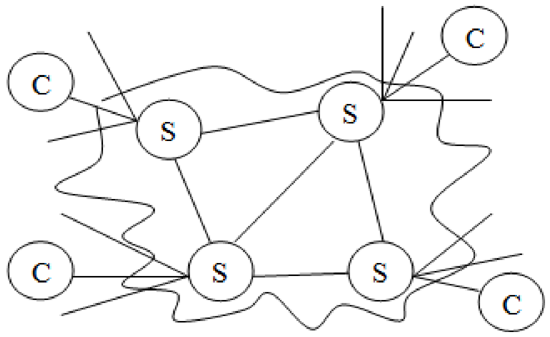

Figure 2. Distributed system with multiple client and server

In this data is stored on multiple storage devices as a replica. Data replication may be synchronous or asynchronous depend upon data consistency. Components are replicated on different machines, and if any component or machine fail, then that application can be transferred and run on another machine having the required components [23].

B. Check-pointing

It is the process to saving from complete execution of a task. It checks the acceptance test, if fail then go to previous checkpoint instead of beginning. A check point may be system level, application level, or mixed level depends on its characteristics [15]. Check-pointing is also categorized on the basis of In-transit or orphan message as shown in figure 3. These are Uncoordinated Checkpointing, Coordinated Check-pointing, and Communication-induced Check-pointing. Check-pointing also can be classified is based on who instruments the application that do the actual capturing and re-establishing of the application execution state. These are Manual code insertion, Pre-compiler check pointing, Post-compiler check-pointing [16], [17].

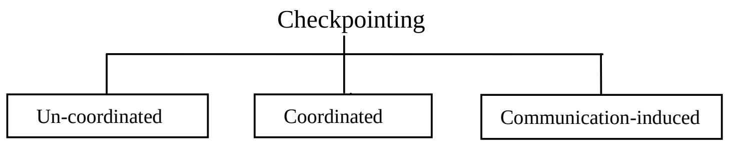

Figure 3. Types of Check-pointing

A check point may be local or global on the basis of their scope. Check-point for separate process is local check-point and a check-point applied for set of processes is called global check-point. Check-pointing have some demerits such as Check-pointing causes execution time overhead even if there are no crashes. The cost of writing check-point data to stable storage whenever a check-point is taken is called the check-pointing cost [26].

C. Scheduling

It is also one of the methods to tolerate fault from distributed system [3], [9]. It is used to overcome the drawback of check-pointing in distributed environment. It is categorized as time-sharing scheduling, space-sharing scheduling, and hybrid combination of both. Scheduling is used for load balancing as well as fault tolerance in distributed system on the basis of space or time sharing [11], [18]. There are three approaches of scheduling such as space, time, and hybrid. Space scheduling is used to tolerate permanent or hardware type of fault from a system. The Primary-Backup approach is applied in space redundancy. Time redundancy is used when there is intermittent type of fault in the system. And hybrid redundancy is used when both are required [12].

There are some important methods for tolerating fault in various systems given by many authors in their research. According to Alain Girault et al. [1] the Algorithm Architecture Adequation (AAA) method will generate a static code automatically for real time distributed embedded system. This method basically used for processor failure with fail stop behavior. There are two main scheduling techniques (offline, online) used in software or hardware failure of a system. Offline is one mostly preferred in real time embedded system [2]. Offline scheduling is preferred because there can be two types of failure: fail-silent and omission which can be tolerated by replication of operations and data communications. In replication the scheduling technique is based under the state-machine and the primary/backup arbitrations between replicas and in communication we assume that different communication uses distinct buffer.
TERCOS and DEBUS are the best approaches used to exploiting redundancies in Fault-Tolerant and Real-Time Distributed Systems [3], [4]. These techniques are based upon off line scheduling for exploiting redundancy in which Tercos can be significantly improve schedulability by up to 17.0% (with an average of 9.7%). and empirical results reveal that Debus can enhance schedulability over Tercos by up to 12% (with an average of 7.8%).
Most of the scheduling algorithms are based on the primary–backup scheme for periodic real-time tasks, introduce unnecessary redundancies by aggressively using active-backup copies but these algorithms are based upon fixed-priority to enhance schedulability. Some of the basic algorithm like RMFF uses the WCRT in which primary copy and backup copies are of same size to check the schedulability of task and determine the status of backup copies executes in passive form. *In Tercos*, the WCRTs of primary copies are always less than or equal to the WCRTs of the corresponding backup copies in which by using best-fit approach we can enhance the schedulability.
The technique known as online scheduling has been explained in MPI-FT to tolerate fault in MPI [5]. In this authors propose the mechanism for detecting process failure and there recovery mechanism in case of failure. In monitoring process, each process keeps a buffer with its own message traffic to be used in case of failure and the

implementer uses periodical test for notification of failure. For recovery all the communications of processes are simulating with dead one and solution is provided for dead process.
Processor and communication link failure can be tolerated by using offline scheduling technique and generate a fault tolerate distributed schedule. In this the algorithm based upon graph transformation can tolerate fixed number of arbitrary processor and communication link failure [6]. The graph is data flow graph shown in figure 4. in which the each vertex is an operation and each edge is data dependence.

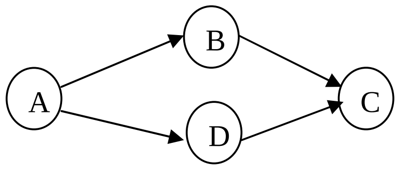

Figure 4. A data dependence graph.

*“Authors propose to use graph transformation to perform software redundancy, where a given input algorithm graph (Alg) is transformed into a new algorithm graph (Alg *) augmented with redundancies. Then, operations and data-dependences of Alg * can be distributed and scheduled on a specified target distributed architecture (Arc) to generate a fault tolerant distributed schedule”* [6].

Tolerating fault in a dynamic grid environment such that link down or resource failure is very frequent. Fault occurs in Hierarchical dynamic scheduler (HDS) for grid workflow applications can be tolerated [8]. In HDC all the resources are arranged in hierarchical tree manner which works as a scheduler. The root failure of the tree can be tolerated by randomization of multiple simultaneous. When the root failure is tolerated then the important thing is to maintain the tree. For that authors start from initial tree with satisfying some properties as fault detection, recovery, (handling child failure and handling parent failure), information coherence (node join, node failure, etc) [7]. Handling of messages in the presence of failures is also there in which different types of massages are handled such as Record path and randomization, Acceptance message, Deadline violation message, Signal execution message and Role of portal and root failure.
As we show the many methods are there for tolerating fault in a grid [5] [6] [7] [8]. But all of these are for link failure, resource failure, etc. Job scheduling is one of the method in grid computing for scheduling a task. The fault can be occur in loosely coupled job scheduling with job replication scheme such that jobs are efficiently and reliably executed can be tolerated [10].

VII. CONCLUSION

To conclude this paper, we can say that it is very different to detect a fault in distributed system as compare with uniprocessor. Fault tolerance techniques are also depending upon its occurrence. In this paper we explore various reliable fault detection and fault tolerance methods. There are three things in real time distributed system which should be kept in mind when fault tolerance is applying. These are reliable, scalable and feasible. In real time distributed system feasibility of a task is much important because there is a dead line defined for every task and the task should be finished on or before its deadline even there is a fault in the system.

VIII. FUTURE WORK

We have seen many fault detection and recovery technique but the very basic question arises in mind, are these reliable? Are these working efficiently in real time scenario? Are these working properly on different architecture? Are two nodes which are not directly connected can detect each other properly? Is the technique which is tolerating fault on real time, feasible? These are the things which can be explored in future research. The work can also be explored for Ad-hoc network.

IX. REFERENCES

[1]. Alain Girault, Christophe Lavarenne, Mihaela Sighireanu, Yves Sorel. Generation of Fault-Tolerant Static Scheduling for Real-Time Distributed Embedded Systems with Multi-Point Links. In IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, San Francisco, USA, April 2001.
[2]. Dima, Alain Girault, Christophe Lavarenne, Yves Sorel, Off-line real-time fault-tolerant scheduling. In Euromicro Workshop on Parallel and Distributed Processing, Mantova, Italy, February 2001.
[3]. Wei Luo, FuMin Yang, Gang Tu, LiPing Pang, Xiao Qin “TERCOS: A Novel Technique for Exploiting redundancies in Fault-Tolerant and Real-Time Distributed Systems” 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications(RTCSA 2007).

[4]. Wei Luo, Xiao Qin, Member, IEEE, Xian-Chun Tan, Ke Qin, and Adam Manzanares “Exploiting Redundancies to enhance Schedulability in Fault-Tolerant and Real-Time Distributed Systems” IEEE Transactions on system man and cybernetics – part A : systems and humans, VOL. 39, NO. 3, May 2009.

[5]. Louca, neophytou. Lachanas. And evripidou “MPI-FT: portable fault tolerance for MPI” In parallel processing latters Vol 10,no. 4 (2000) pg.371-382.

[6]. Alain Girault, Hamoudi Kalla, and Yves Sorel “An active replication scheme that tolerates failure in distributed embedded real time system”

[7]. Nitin B. Gorde, Sanjeev K. Aggarwal “A Fault Tolerance Scheme for Hierarchical Dynamic Schedulers in Grids” IEEE international Conference on Parallel Processing – Workshops -2008

[8]. Nitin B. Gorde and Sanjeev K. Aggarwal. “Hierarchical dynamic scheduler for grid workflow applications”. Technical Report, CSE, IIT Kanpur India, TRCS-2008-258, Source
url: http://www.cse.iitk.ac.in/users/niting/HDS.pdf.

[9]. J. H. Abawajy “Fault-Tolerant Scheduling Policy for Grid Computing Systems” Proceedings of the IEEE 18th International Parallel and Distributed Processing Symposium (IPDPS’04).

[10]. Paul Townend, Jie Xu,” Fault tolerance within a grid Environment”, As part of the e-Demand project at the University of Durham, DH1 3LE, United Kingdom, 2003.

[11]. VINCENZO DE FLORIO and CHRIS BLONDIA “A Survey of Linguistic Structures for Application-Level Fault Tolerance” ACM Computing Surveys, Vol. 40, No. 2, Article 6, Publication date: April 2008.

[12]. Alain Girault, Hamoudi Kalla, and Yves Sorel “TRANSIENT PROCESSOR/BUS FAULT TOLERANCE FOR EMBEDDED SYSTEMS, with hybrid redundancy and data fragmentation”

[13]. Naveed Arshad, A planning-based approach to failure recovery in distributed systems, A thesis submitted to the University of Colorado in partial fulfilment of the requirements for the degree of Ph.D., 2006.

[14]. Paul Townend, Jie Xu, Replication-based fault tolerance in a grid environment, As part of the e-Demand project at University of Leeds, Leeds, LS2 9JT, UK, 2004.

[15]. Greg Bronevetsky, Daniel Marques, Keshav Pingali, Paul Stodghill, Automated Application-level Checkpointing of MPI Programs, Cornell University, Ithaca, NY, 2003, pp. 84.94.

[16]. Sriram Krishnan, An architecture for check pointing and migration of distributed components on the Grid, A thesis submitted to Indiana University in partial fulfilment of the requirements for the degree of Doctor of Philosophy, 2004.

[17]. Pawel Garbacki, Bartosz Biskupski, Henri Bal3, Transparent fault tolerance for grid applications, University of Technol-ogy, Delft, The Netherlands, 2005.

[18]. Sriram Krishnan, Dennis Gannon, Checkpoint and restart for distributed components in XCAT3, in: Proceedings of the Fifth IEEE/ACM InternationalWorkshop on Grid Comp. GRID, 2004.

[19]. P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski. A fault detection service for wide area distributed computations. In Proc. 7th IEEE Symp. on High Performance Distributed Computing, pages 268-278, July 1998.

[20]. R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In N. Davies, K. Raymond, and J. Seitz, editors, Middleware 1998, pages 55-70, The Lake District, UK, 1998.

[21]. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. Journal of the ACM, 43(2): 225-267, Mar. 1996.

[22]. M.K. Aguilera, W. Chen, S. Toueg, Heartbeat: A time- outfree failure detector for quiescent reliable com- munications, in: Proceedings of the 11th International Workshop on Distributed Algorithms, WDAG.97, September 1997, pp. 126.140.

[23]. Gosia Wrzesin´ ska, Rob V. van Nieuwpoort, Jason Maassen, Henri E. Bal, Fault-tolerance, malleability and migration for divide-and-conquer applications on the grid, in: Proceedings of International Parallel and Distributed Processing Sympo- sium . IEEE, 2005.

[24]. Amit Jain, R.K. Shyamasundar, Failure detection and mem-bership management in grid environments, in: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Com-puting .GRID-2004, pp. 44.52.

[25]. I. Gupta, T.D. Chandra, G.S. Goldszmidt, On scalable and efficient distributed failure detectors. in: 20th ACM Symposium, On Principles of Distributed Computing, Newport, RI, August, 2001.

[26]. S. Siva Sathya, S. Kuppuswami, K. Syam Babu, Fault tolerance by check-pointing mechanisms in grid computing, in: Proceedings of the International Conference on Global Software Development, Coimbatore, 26.28 July 2007.

X. AUTHORS PROFILE

Arvind Kumar: Mr. Arvind Kumar is presently pursuing M. Tech. (Computer Science and Engineering) from Motilal Nehru National Institutre of Technology, Allahabad, U.P., India. He has received B. Tech. degree from UPTU, Lucknow in Computer Science and Engineering. He has worked as a Lecturer in ABES Institute of Technology, Ghaziabad. He has more then 8 papers in National/International conferences. His area of interest is Real-Time System, Fault Tolerance, and Theory of computation.

Rama Shankar Yadav: Dr. Rama Shanka Yadav is currently Professor at Motilal Nehru National Institute of Technology, Allahabad. He received Ph.D. degree from IIT, Roorkee, M.S. degree from B.I.T.S. Pilani and B.Tech degree from I.E.T, Lucknow. Dr. Yadav has extensive research and academic experience. He had woked in leading institutions such as GBPEC, Pauri Garhwal, BITS, Pilani. He has authored more than 50 research papers in National/International conference and referred Journals/Book chapters. Dr. Yadav’s areas of interest are Real-Time System, Embedded System, Fault Tolerant System, Energy Aware Scheduling, Computer Architecture, Distributed Computing and Cryptography.

Ranvijay: Mr. Ranvijay is presently pursuing Ph.D from Motilal Nehru National Institute of Technology Allahabad. He has received B. Tech. degree from Purvanchal University and M.Tech in computer science and Engineeirng form Motilal Nehru National Institute of Technology Allahabad,India. He has worked as Lecturer at a Institute of Northern India Engineering College Lucknow. He has authored more than 12 research papers in international conference and reputed journal/book chapters. Mr. Ranvijay's areas of interest are Real-Time System, Computer Architecture, Energy Aware Scheduling and Mobile Computing.

Anjali Jain: Ms. Anjali Jain is presently pursuing M. Tech. (Computer Science and Engineering) from Motilal Nehru National Institutre of Technology, Allahabad, U.P., India. She has received B. Tech. degree from UPTU, Lucknow in Computer Science and Engineering. She has worked as a lecturer in ABES Institute of Technology, Ghaziabad. Her area of interest is Real-Time System, Fault Tolerance and Mobile Computing. Her programming interest concerned in JAVA and C.