Developing Fault Tolerance Integrity Protocol for Distributed Real Time Systems

Dhuha Basheer Abdullah **Amira B. Sallow**
Prof.dhuha_basheer@uomosul.edu.iq
College of Computer Sciences and Mathematics
University of Mosul, Mosul, Iraq

Received on: 21/10/2012 **Accepted on: 30/01/2013**

ABSTRACT

In the distributed real time systems, tasks must meet their deadline even in the presence of hardware/software faults. Fault tolerance in distributed real time systems refers to the ability of the system to meet the tasks deadline and to detect their failure and recover them. In this paper, we considered the problem of fault tolerance and developed a fault tolerance protocol called DRT-FTIP (Distributed Real Time – Fault Tolerance Integrity Protocol).This protocol increases the integrity of the scheduling in distributed real time systems.

Keywords: Real Time, fault Tolerance, Scheduling, Protocol

تطوير بروتوكول متكامل كاشف للأخطاء في أنظمة الزمن الحقيقي الموزعة

ضحى بشير عبد الله **اميرة سلو**
كلية علوم الحاسوب والرياضيات، جامعة الموصل

تاريخ استلام البحث: 2012\10\21 **تاريخ قبول البحث: 2013\01\30**

الملخص

في أنظمة الوقت الحقيقي الموزعة المهام من الضروري أن تنجز في وقتها المحدد حتى وان كان هنالك بعضاً من الأخطاء المادية أو البرمجية. التحكم بالخطأ في أنظمة الزمن الحقيقي الموزعة يعني قدرة النظام على تحقيق الحدود النهائية للمهام واكتشاف الأخطاء وتجاوزها. في هذا البحث تم الأخذ بنظر الاعتبار مشكلة التحكم بالأخطاء وتم تطوير بروتوكول التحكم بالخطأ سمي بـ(بروتوكول التحكم بالخطأ المتكامل لأنظمة الزمن الحقيقي الموزعة). عمل البروتوكول المقترح على زيادة تكامل الجدولة لأنظمة الزمن الحقيقي الموزعة.

الكلمات المفتاحية: الزمن الحقيقي، التحكم بالخطأ، الجدولة، بروتوكول

1. Introduction

Real-time systems can be classified as hard real time systems in which the consequences of missing a deadline can be catastrophic and soft real time systems in which the consequences are relatively tolerable. In hard real time systems, it is important that tasks complete within their deadline even in the presence of a failure. Examples of hard real-time systems are control systems in space stations, auto pilot systems and monitoring systems for patients with critical conditions. In soft real-time systems, it is more important to economically detect a fault as soon as possible rather than to mask a fault. Examples of soft real-time systems are all kinds of airline reservation, banking, and E-commerce applications[3].

A faulty system, due to any reason during processing some task, can cause 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[4].

1.1 Contributions

In this paper, we consider the problem of Fault tolerance in distributed real-time systems. We design a Distributed Real Time – Fault Tolerance Integrity Protocol (DRT-FTIP) that has the following prosperities:

1. Designed to function in dynamic network.
2. Coupled with EOE-DRTSA (End to End-Distributed Real Time Scheduling Algorithm.
3. Control integrity and fault tolerant with proposed Distributed real-time system.

1.2 Related Work

Past works on developing fault tolerance integrity protocol by Edward Curley, Binoy Ravindran, Jonathan Anderson, and E. Douglas Jensen who considered the problem of recovering from failures of distributable threads in distributed real-time systems that operate under run-time uncertainties including those on thread execution times, thread arrivals, and node failure occurrences. They presented thread integrity protocol called TPR[4].

Binoy Ravindran, Edward Curley, Jonathan Anderson, and E. Douglas Jensen who considered the problem of recovering from failures of distributable threads in distributed real-time systems. They presented a scheduling algorithm called HUA and two thread integrity protocols called D-TPR and W-TPR [5].

2. Failure, Error, and Faults

Avizienis and others define the terms failure, error and faults as fellows [6]:

Failure: A failure system occurs when the service provided by the system deviates from the specified service. For example, when a user cannot read his/her stored file from computer memory, then the expected service is not provided by the system.

Error: An error is a perturbation of internal state of the system that may lead to failure. A failure occurs when the erroneous state causes an incorrect service to be delivered, for example, when certain portion of the computer
memory is corrupted or broken and stored files, therefore cannot be read by the user.

Fault: The cause of the error is called a fault. An active fault leads to an error; otherwise the fault is dormant. For example, impurities in the semiconductor devices may cause computer memory in the long run to behave unpredictably.

3. Techniques for Fault Tolerance

Fault tolerance is the ability to continue operating despite the failure of a limited subset of their hardware or software. So, the goal of the system designer is to ensure that the probability of system failure is acceptably small. There can be either hardware fault or software fault, which disturbs the real time systems to meet their deadlines. There are three types of faults: Permanent, intermittent, and transient.

A permanent fault does not die away with time, but remains until it is repaired as the affected unit is replaced. This is an intermittent fault cycle between the fault–active and fault benign states. A transient fault dies away after some time[4].There are different types of fault which can occur in Real-Time Distributed System. These faults can be classified depending on several factors such as:

Network fault: A Fault occurrence in a network is 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 occurrence is due to media head crashes.

Processor faults: fault occurs in processor due to operating system crashes.

Process faults: The occurrence of fault is due to shortage of resource, software bugs.

Service expiry fault: The service time of a resource may expire, while application is using it.

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. Faults occur with respect to time are shown in Figure 1 below:

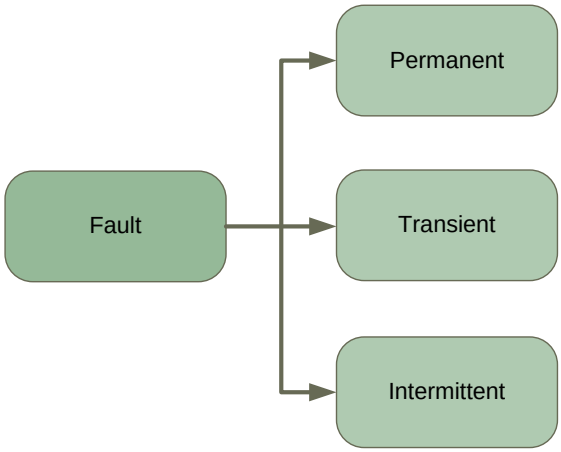

Figure 1. Types of Fault in a System

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 which appear 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 [4].

4. The System Model

We consider a distributed system architecture model [1] consisting of heterogeneous processors; a set of client nodes; and a set of server nodes. These nodes were interconnected via a communication network. A single Global Scheduler for the

system was responsible for computing the initial priorities for the tasks; scheduler utility was checked based on information provided by the application programmer. As new tasks were introduced to the system, the Global Scheduler distributes them to client processors. A Local Scheduler in each processor was responsible for specifying the order of threads executions on a node. Thus, scheduling decisions made by a node scheduler are independent of that made by other node schedulers. Client Node schedulers make scheduling decisions using thread scheduling attributes, which typically include threads' time constraints (e.g., importance, urgency). When a new job arrives at client node and its utility is larger than the utility of the currently executing job, the currently job will be preempted and the new job will be scheduled for execution.

EOE-DRTSA algorithm [1] was designed to overcome the shortcomings of independent scheduling algorithms by taking into account global information, while constructing schedule. In EOE-DRTSA scheduling, firstly the thread will be arrived to the Global Scheduler server. The Global Scheduler will compute the initial Urgency and DTUF value. We consider the DTUF (Developed TUF) function to be three dimensional function that decouples importance and urgency of a thread, urgency is measured on the X-axis, and importance is measured on the Y-axis. The Benefit is denoted by DTUF and is measured on the Z-axis. After the DTUF values were computed for the threads, the Global Scheduler will pass the threads to the target node to be executed there and the DRT-FTIP protocol will be worked concurrently with the EOE-DRTSA scheduling algorithm. When the thread arrived to the Local Scheduler node, the node will schedule the thread depending on its local scheduling algorithm. DRT-FTIP integrity protocol will monitor the scheduling work on the local scheduler.

5. Protocol Algorithm Description

We design an DRT-FTIP(Distributed Real Time – Fault Tolerance Integrity Protocol) with **EOE-DRTSA** scheduling algorithm for distributed real time system. This protocol will control the faults that occurred in this proposed system. In server side, when a thread sent to the client the server will expect to receive (thread execute message) from the client. This message means that the thread is executed successfully and there is no error. The DRT-FTIP protocol has three operation phases:

Phase I:

Establish the protocol for the first time, the server will send *Client_Hello* message and wait until the client responds with the message *Server_Hello.* After that, the second phase will begin.

Phase II:

In this phase, there are two states of behavior:

a) Normal state behavior: This behavior will describe the successful work for the DRT-FTIP protocol without any error. In server side, for each thread that has been sent to the client, the server will send thread *Execute_wait* message and commit *wait* state in the state field in the *Execute_Table* that belongs to the sender thread. When the *Execute_finish* message arrived from the client the server changes the state field in the *Execute_Table* to *finished* to point that the thread was executed successfully. Figure(2) shows the normal behavior of DRT-FTIP integrity protocol.

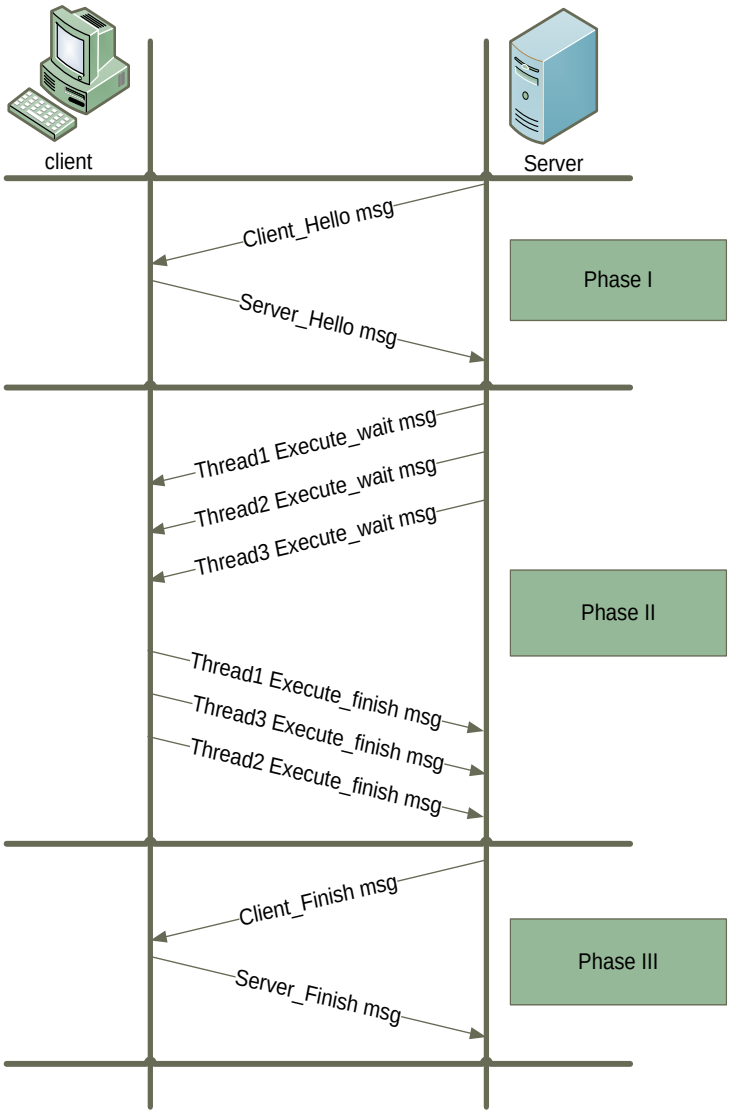

Figure 2. DRT-FTIP Integrity Protocol: Normal State Behavior

b) Anomaly state behavior: This behavior describes unusual work of DRT-FTIP protocol. The server check the Execute_Table every Tp (time period). If an error will be occurred in client side the client system will send exception to the client application. Actually, exceptions were used for handling unusual errors in programs that will be occurred during the program execution. The exception handling mechanism was used to provide a means to detect and report exceptional circumstances so that, an appropriate action will be taken. The mechanism suggests incorporation of separate error handling code that performs the following tasks:

   a. Find the problem
   b. Inform that an error has been occurred
   c. Receive the error information

   For any exceptions occurred, the client exception handling will send error_message to the server explaining the error type. If the error was caused by a thread, the server will try to send the thread and Thread Execute_wait message again for three times until Execute_finish message will be arrived. If the error was caused by the client machine, the server may eliminate that thread from the system

or send it to another client and begins the same operations again. Figure(3) shows the anomaly behavior for DRT-FTIP integrity protocol.

Phase III:

If there are no threads in the ready queue in server side or the server wants to finish the protocol in a client, the server will send Client_finish messag and will recive Server_finish message from the client.

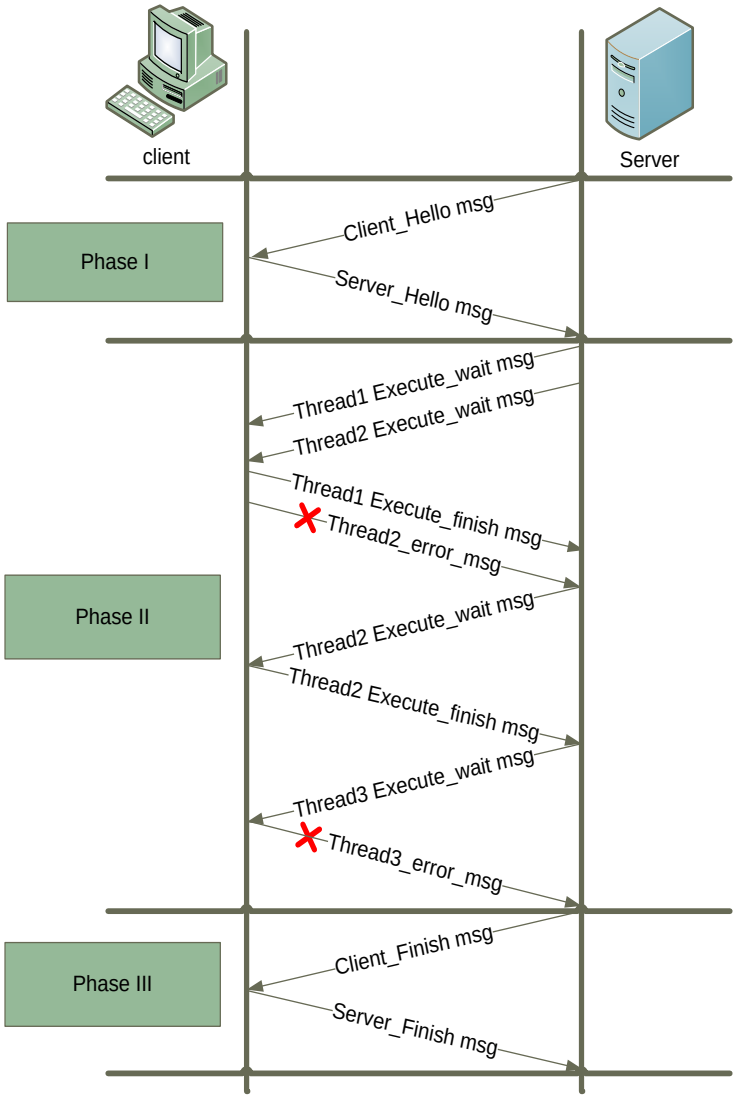

Figure 3. DRT-FTIP Integrity Protocol: Anomaly State Behavior

6. Conclusion

In this paper, we developed a fault tolerance integrity protocol for distributed real-Time systems. The advantage of the algorithm is that the threads considered in this system are dynamic and aperiodic. The proposed protocol was simple and easy to implement. The proposed protocol increases the integrity of the scheduling algorithm used with it.

REFERENCES

[1] Amira Bibo, Dhuha Basheer, 2012, "EOE-DRTSA: End-to-End Distributed Real-time System Scheduling Algorithm", Dept. of Computer Science College of Computer Sciences and Mathematics University of Mosul, under publishing.

[2] Arvind Kumar, Rama Shankar Yadav, Ranvijay, Anjali Jain, 2011, "Fault Tolerance in Real Time Distributed System", Department of Computer Science and Engineering Motilal Nehru National Institute of Technology, Allahabad.

[3] Christy Persya, T.R.Gopalakrishnan Nair, 2008, "Fault Tolerant Real Time Systems", Department of Information Science and Engineering, The Oxford College of Engineering, Bangalore, India.

[4] Edward Curley, Binoy Ravindran, Jonathan Anderson, E. Douglas Jensen, 2007, "Recovering from Distributable Thread Failures in Distributed Real-Time Java", Virginia Tech Blacksburg, USA.

[5] Edward Curley, Binoy Ravindran, Jonathan Anderson, E. Douglas Jensen, 2006, "Integrity Protocols for Recovering from Distributable Real-Time Thread Failures with Assured Timeliness in Dynamic Systems", Virginia Tech Blacksburg, USA.

[6] Risat Mahmudpathan, 2010, "Scheduling Algorithms For Fault-Tolerant Real-Time Systems", Department of Computer Science and Engineering, Chalmers University of Technology, Goteborg, Sweden.