Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 87 (2016) 50 – 54

Procedia Computer Science

Fourth International Conference on Recent Trends in Computer Science & Engineering

A Study on Byzantine Fault Tolerance Methods in Distributed Networks

M A Nasreen[a]*, Amal Ganesh[b], Sunitha C[c]

[a]*PG Scholar, Vidya Academy of Science & Technology, Thrissur, 680501, India*
[b]*Asst. Professor, Vidya Academy of Science & Technology, Thrissur, 680501, India*
[c]*Asso. Professor, Vidya Academy of Science & Technology, Thrissur, 680501, India*

Abstract

With the evolution of new technologies, the growing reliance on networks has increased immensely. Such an advancement induces malicious attacks and software errors to occur frequently. Building robust network services that can tolerate a wide range of failure types is a fundamental problem in distributed systems. The most fundamental approach, called Byzantine fault tolerance, helps mask arbitrary failures exhibited by failing nodes. Here, the problem of reliably broadcasting messages in a mutli hop network is dealt, where some nodes are likely to fail. Most of the existing solutions require high network connectivity and these requirements become difficult to satisfy when the network grows large. In this paper, a study on various Byzantine fault tolerant methods has been carried out which has been developed and implemented by research experts in this field. These solutions are particularly customized to adapt to sparsely connected networks.

Keywords: Byzantine Failures; Multi hop Network; Asynchronous Network;, Fault Tolerance; Reliable Broadcast; Distributed Computing

1. Introduction

In this modern era, the World Wide Web is being used by millions of people on a day-to-day basis for various purposes. Through the use of a standard web browser, the user can acquire information stored on Web servers situated anywhere on the globe. This results in an illusion that all this information is situated locally on the user's computer. But in reality, the Web depicts a huge distributed system that occur as a single resource to the user.

There are several definitions and outlooks on what distributed systems are. According to Tanenbaum, a distributed system is interpreted as "A group of independent computers that portrays as a single computer to the users of the system" [2]. Leslie Lamport's interpretation of distributed system is postulated as "A distributed system is one in which the failure of a computer you didn't even know existed can convert your own computer unusable".

A common misapprehension that surrounds the concept of distributed systems is that it is just another label for a network of computers. However, this neglects an important distinction. A distributed system is one that is built on top of a network and it tries to hide the existence of multiple autonomous computers. A distributed system appears as a single entity providing the user with the services required.

* Corresponding author. Tel.:+91-9539750604
E-mail address:nasreen23ma@gmail.com

Peer-review under responsibility of the Organizing Committee of ICRTCSE 2016
doi:10.1016/j.procs.2016.05.125

One of the compelling characteristic of distributed systems that sets them apart from single machine systems is the notion of partial failure. The terminology partial failure is used when a part of the system is failing while the remaining part continues to operate, and seemingly correctly. Hence, an important objective in distributed systems design is to construct system in such a way that it can automatically recover from partial failures without sorely affecting the overall performance. In other words, a distributed system is expected to be fault tolerant. While designing any system to be fault tolerant, it first requires the selection of a fault model, a set of probable failure scenarios along with an intellect of the frequency, duration, and impact of each scenario.

A wide range of communication applications assume the availability of a reliable network. At this stage, data are expected to traverse the network and to arrive intact at their destination. Therefore, achieving fault tolerance and reliable communication are strongly related.

2. Fault Tolerance- An Overview

Fault tolerance is one which is strongly related to dependable systems. Dependability covers a number of favorable requirements for distributed systems including the following: availability, maintainability reliability, and safety.

Availability refers to the property in which a system is ready to be used immediately. In particular, it is defined as the probability the system is operating correctly at any given moment and is able to perform functions on behalf of its users. Reliability is defined as the property that a system can run continuously without failure. Safety is defined as the situation that when a system temporarily fails to operate correctly, nothing catastrophic happens. Finally, maintainability pertains to how easily a failed system can be restored. However, automatic recovering from failures is much harder in practice than in theory. A system is said to have failed when it is unable to meet its promises. Particularly, if a distributed system is designed to provide its users with a certain number of services, the system is said to have failed when some of those services cannot be provided. An error is described as a part of the system's state that may lead to a failure.

Faults are classified as transient, intermittent or permanent. Transient faults occur only once and then tends to disappear. If the operation is repeated, the fault tends to disappear. An intermittent fault occurs, then vanishes, then recur and so on. Intermittent faults are caused by loose wires. Finally, a permanent fault is one that persists until the faulty component is replaced. Dependable systems are required to control faults. It is important to comprehend how to handle failure when one has occurred. Table I helps understand the various types of failure schemes which are described by Cristian [3] and Hadzilacos and Toueg [4].

Table I. Different types of failure

Type of failure	***Description***
Crash Failure	A server is working correctly until it halted.
Omission Failure • Receive Omission • Send Omission	A server fails to respond to incoming requests. A server fails to receive incoming messages. A server fails to send messages.
Timing Failure	A server's response lies outside the time interval.
Response Failure • Value Failure • State Transition Failure	A server's response is incorrect. The value of the response is wrong. The server deviates from correct flow of control.
Arbitrary Failure	A server produce arbitrary response at any given time.

A crash failure is said to have occurred when a server prematurely halts, but was working correctly until it stopped. An omission failure occurs when a server fails to respond to a request. In the case of a receive omission failure, there is a possibility that the server by no means got the request in the first place. Generally, a receive omission failure will not influence the current state of the server, as the server is oblivious of any message sent to it.