A STUDY ON THE FAULT TOLERANCE IN IOT

Mulamreddy Subba Reddy
Research Scholar,
North East Frontier Technical University, Arunachal Pradesh (India).
Dr. Vijay Pal Singh
HoD - (Computer Science),
Sun Shine College, Mandrella, Jhunjhunu.

Abstract
The term "Internet of Things" (IoT) has a comprehensive definition that encompasses a variety of applications. It expands the traditional Internet notion to include interconnected items that form pervasive computing environments, in which a variety of heterogeneous sensor devices are linked with the goal of mining the data they generate. Previously, communication channels involved human-to-human or human-to-device communication, but with the growth of the Internet of Things (IoT), machine-to-machine communication has emerged. It means that one machine can talk with another machine via the internet and exchange data wirelessly. In today's world, the Internet of Things (IoT) is a worldwide computing network in which everything is connected to the internet. From a research standpoint, the Internet of Things offers an endless number of possibilities. Fault-tolerant control systems are closed-loop systems that can maintain good performance despite uncertainty. The incidence of a component defect and exogenous disturbances, for example, are sources of uncertainty. In both theoretical research and industry applications, fault-tolerant design challenges in dynamic systems have gotten a lot of attention.

Keywords: Fault Tolerance, IOT, ITU

Introduction

ITU and IERC describe the Internet of Things (IoT) as an evolving global network infrastructure with self-configuring abilities following established and inter - operable communication protocols, in which physical and digital "things" have identities, physical traits, and virtual personalities, utilise integrated sensors, and are managed as a single integrated network. IoT has progressed from a futuristic approach - with a fair amount of hype - to a growing market reality in the last year.

Major ICT companies such as Google, Apple, and Cisco have made substantial strategic decisions in order to place themselves in the IoT landscape. Machine-to-machine (M2M) and the Internet of Things (IoT) have become a core business focus for telecom operators, with significant increases in the number of connected objects in their systems. Manufacturers of wearable devices, for example, anticipate a whole new business segment as the IoT becomes more widely adopted.

Embedded systems and cyber-physical systems, network technologies, semantic interoperability, operating platforms and security, and generic enablers are among the areas where the EU has already made investments for some time in funding research and advancement in the field of IoT. These findings are now feeding into advancement, and there are a number of components available that could be usefully exploited and improved by the market.

As a result of this trend, the majority of governments in Europe, Asia, and the Americas regard the Internet of Things as a source of innovation and growth. Although some larger players in some application areas are still unaware of the potential, many others are paying close attention and even speeding up the process by coining new terms for the Internet of Things and introducing additional components to it. Furthermore, end-users in both the private and business sectors have developed significant expertise in dealing with connected devices and networked applications.

As the Internet of Things develops, a combination of related technology approaches and concepts such as Cloud computing, Future Internet, Big Data, Robotics, and Semantic technologies will be used to approximate future potential. Of course, the technology is not new in and of itself, but since some of these ideas crossover (technical and service architectures, virtualization, interoperability, automation), true innovators focus on complementarity rather than protecting individual domains.

Literature Review

N. Mohamed (2019) discusses the various building blocks for providing resilient IoT services in the fog for smart city applications. They suggest connecting IoT devices to redundant fog nodes in order to survive fog node failures. They propose that for better fault tolerance, fog node communication coverage should overlap across numerous IoT distribution areas, and fog nodes should supervise each other based on proximity for quicker failure recovery. The authors also highlight the importance of storing state data for stateful fog services in smart city applications. The discussed solution for dealing with failures in this context is based on replication of stateful fog services, in which replicas of a fog service are updated with each service invocation as well as change in state data, allowing a replica to replace a failed service. However, service replication in a Fog-IoT environment may not always be possible because a service can be linked to an IoT device and thus cannot be replicated without also replicating the device. Replication of stateful IoT devices is also unrealistic, if not impossible. For example, updating the state of an actuator's replica infringes consistency with respect to the PW because actions are performed multiple times in the PW. Geographical constraints may prevent replication from being used: some IoT devices, such as smart window blinds, can only operate in a single space and thus cannot support physical replicas. When replication is possible, it may also be prohibitively expensive.

Tuan (2015) proposes a healthcare-specific architecture for supporting network fault tolerance. For an energy-efficient communication infrastructure, it is premised on the wireless protocol 6LoWPAN. The architecture is made up of customizable star-based 6LoWPAN sensor nodes that retrieve patients' bio-signals and are linked to a gateway made up of multiple sink nodes with backup routing and internet access. To provide fault tolerance, the inactivity of a sensor node for a predefined period of time stimulates a discovery protocol, which activates actions to determine whether or not a failure has occurred. The protocol starts by requesting the sensor node's status. If the latter does not respond, a warning message is broadcasted via another sink node to rule out the possibility of a faulty sink node. The set of sensor nodes responds to the warning message in order to identify the failure. The use of backup routing between sink nodes allows one to maintain connectivity in the event of a failed connection and avoids traffic bottlenecks caused by high receiving data rates. An alert mechanism for caregivers/doctors is also included in the proposed design. This method employs star-based architecture and also on custom hardware. This method is use case specific and cannot be applied to more general IoT domain applications. Furthermore, to confirm and correct a failure, the approach heavily relies on notification and intervention of caregivers/doctors. In our approach, we aim for full automation of failure detection and recovery.

S. Zhou (2015) took a novel approach to hardware redundancy by employing sensors of various modalities as backups for failed sensors. The approach used regression analysis to identify compatibilities between different sensors in order to determine whether it was possible to integrate data from multiple different sensors into 'virtual services.' When the primary passive infrared (PIR) sensor fails, a microphone in a quiet room could be used to detect intruders. The researchers compared a linear and non-linear model to see which performed better with various modalities. Ten devices with light, PIR, Kinect (Z. Zhang (2012)), sound, and ultrasonic sensors were used.

P. Su (2014) investigates the issue of fault recovery in IoT using backups. It is assumed that multiple replicated services have already been installed in the system. T. N. Gia (2015) reports on another work on fault tolerant health monitoring that uses redundant devices as well as implements an enhanced gateway for fault tolerance. However, in IoT, replicated services are extremely limited. Even if they are of the same modality, sensors deployed in different locations may provide completely different services. To accomplish fault tolerance, we try to avoid using redundant IoT resources.

Definition of Internet of Things

Gartner identified ten "critical" IT trends and technologies for the next five years, including the Internet of Things. Every single one of these items has an IP address and can be tracked. The Internet is infiltrating corporate assets as well as consumer goods such as automobiles and televisions. The issue is that most businesses and technology providers have yet to investigate the possibilities of a larger Internet and are not functionally or organizationally prepared. According to Gartner (2014), there are four basic usage models that are gaining traction:

- Manage,
- monetize,
- operate, and
- expand.

People, things, information, and places can all benefit from these, so the "Internet of Things" will be replaced by the "Internet of Everything." The concept of IP network convergence is fundamental in this context, and it is based on the use of a prevalent multi-service IP network that can support a range of applications and services.

The use of IP to interact with and control small devices such as sensors allows large, IT-oriented networks to be combined with real-time and specialized network services.

Fault Tolerance

Computing systems are made up of a plethora of hardware and software components that can fail unexpectedly. Unexpected failures in many systems can drastically alter the system and lead to service outages. The most common approach to providing failure resiliency (also known as fault tolerance) is to create the system in such a way that when some components makes mistakes and suffer losses, the standby components instantly replace them with no interruption in service to the client . The major characteristic for improving system efficiency when a fault occurs in the system is fault tolerance. This fault tolerance is required because faults can eventuate in hardware but can be considered acceptable by software. The grid computing technique requires the administrator to keep track of control points, replication, and scheduling, among other things. These are the terms that must be used in grid computing software. When designing a fault - tolerance system, some software requirements must be met in order for the system to be tolerated. . The most common approach to providing failure resiliency (also known as fault tolerance) is to create the system in such a way when some components makes mistakes and suffer losses, the standby components instantly replace them with no interruption in service to the client . The major characteristic for improving system efficiency when a fault occurs in the system is fault tolerance. This fault tolerance is required because faults can eventuate in hardware but can be considered acceptable by software. The grid computing technique requires the administrator to keep track of control points, replication, and scheduling, among other things. These are the terms that must be used in grid computing software. When designing a fault - tolerance system, some software requirements must be met in order for the system to be tolerated.

The effects of failures on a process/channel along with its description for the commonly rousing class one and class two failures in a distributed network are Outlined in the tables below. There are so many steps that we have to perform to do fault tolerance as follows:

1. Detect the fault
2. Diagnose the fault
3. Restraint the fault
4. Veiling the fault
5. Remunerate the fault
6. Repair the fault

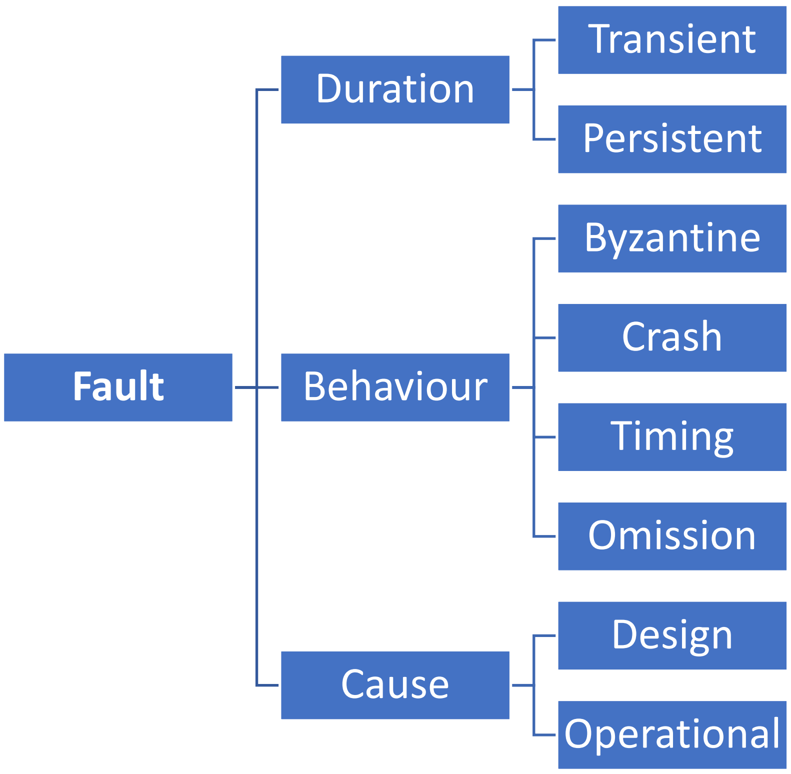
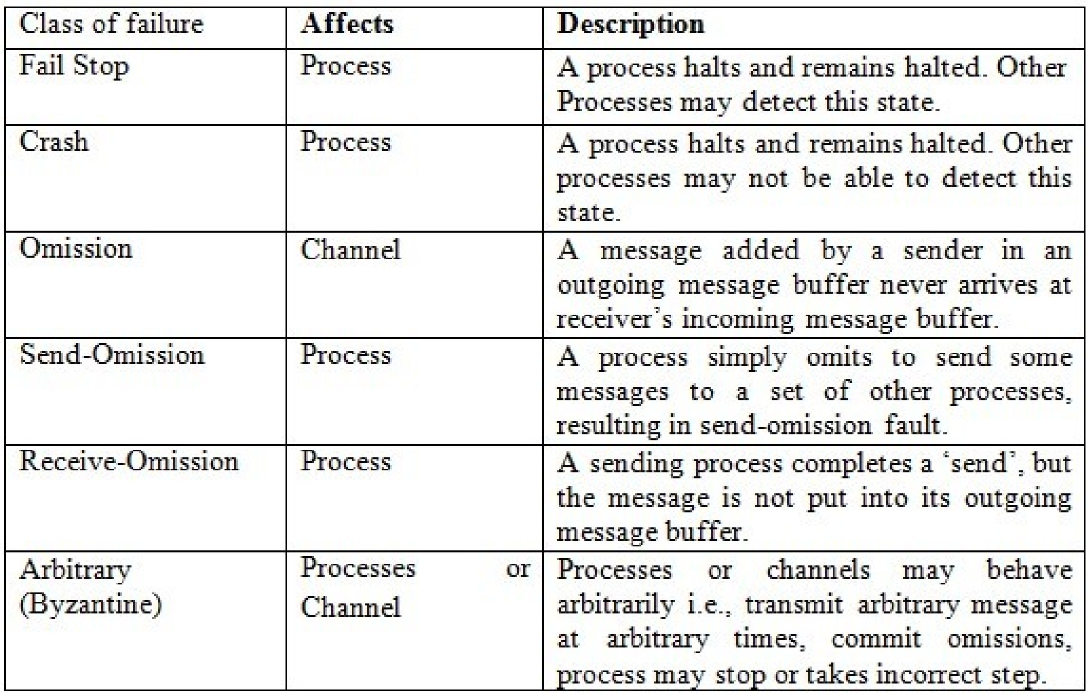


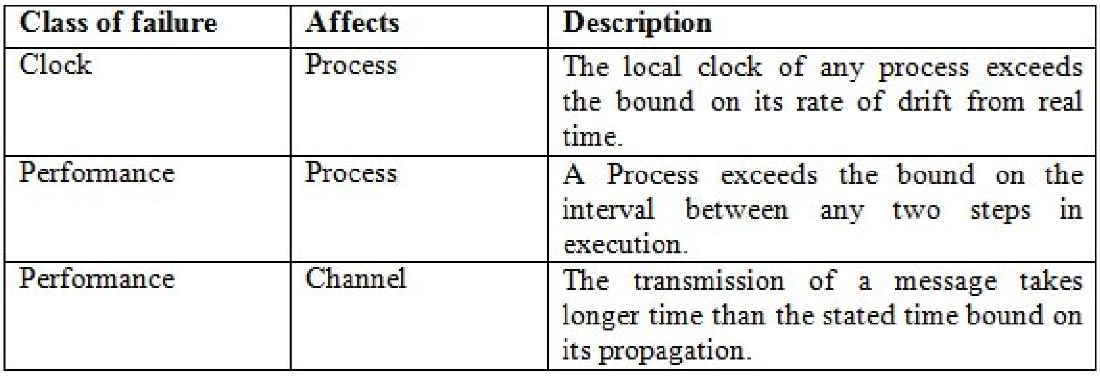
Figure 1: Classification of Fault



Class of failure	**Affects**	**Description**
Fail Stop	Process	A process halts and remains halted. Other Processes may detect this state.
Crash	Process	A process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message added by a sender in an outgoing message buffer never arrives at receiver's incoming message buffer.
Send-Omission	Process	A process simply omits to send some messages to a set of other processes, resulting in send-omission fault.
Receive-Omission	Process	A sending process completes a 'send', but the message is not put into its outgoing message buffer.
Arbitrary (Byzantine)	Processes or Channel	Processes or channels may behave arbitrarily i.e., transmit arbitrary message at arbitrary times, commit omissions, process may stop or takes incorrect step.

Table 1: Class-1 Failure

Table 2: Class -2 Failure



Class of failure	**Affects**	**Description**
Clock	Process	The local clock of any process exceeds the bound on its rate of drift from real time.
Performance	Process	A Process exceeds the bound on the interval between any two steps in execution.
Performance	Channel	The transmission of a message takes longer time than the stated time bound on its propagation.

Methods of finding Faults

There are several methods used for the administration which are described as follows:

Prior And Post Findings

Findings of flaws can be divided into two parts using this method. The first is prior finding, and the second is post finding. Prior discovery, as the name implies, considers failure in advance or before assigning jobs to machines. Failures will be discovered in post findings after they occur. As a result, post findings can only be used in dynamic systems.

The Passive And Active States

This method detects and locates faults in the system in two ways: active and passive states. In an active state, the machines send a heartbeat message to the hardware detector of failure on a regular basis, and the detector sends an acknowledgement with each message. If any machine's acknowledgement fails, the detector will determine the source of the failure. In the passive state, the detector periodically inquires about the presence of each machine.

Representative

In the passive state, the detector periodically inquires about the presence of each machine.

These representatives use this method to monitor the faults in each machine on a regular basis. They also produce a ledger (log) for the hardware required, memory utilisation, resource needs, and time of failures in machine components. These ledgers may be useful in the future to improve the system's reliability and efficiency.

Relevance

Because the system is in operation, if a fault occurs, the system must wait. Because the system contains rapidly changing large computations, relevance-based fault tolerance schemes are preferable. This relevance-based system can achieve high reliability. This is superior to system-based fault tolerance because the system is given long processes. As a result, check pointing is required, among other things.

One framework aids in the deployment of check pointing in a relevance-based fault tolerance plan. It is also in charge of:

- Beginning job submission and detection.
- Conceptual model for administration of check pointing
- Resubmit the faulty jobs that were halted in the interim.

Importance of Faults

- The presence of faults in the system is a common consideration in the application part of any network. The rate of failure will always vary greatly depending on the environment and the size and cost of the devices used. This is due to the following factors:
- The sensors will always be investigated in uncontrollable scenarios. For example, habitat monitoring and surveillance, sewage control, and exploitation in war zones are all examples of unfriendly environments.
- The structure protocols may, in turn, require that nodes be shut down on a regular basis. It will be done to ensure that energy is stored and to help extend the life of those networks.
- Each device's size may be as little as dust particles.

Large-scale applications typically require large numbers of untethered nodes spread across the network. As a design choice, sensor networks typically consist of low-cost equipment to reduce overall costs. Similarly, the energy source of sensor nodes is not rechargeable. The idea is to use redundancy to improve the system's overall fault tolerance while keeping network costs low.

These features, however, make the sensor nodes more prone to failure. As a result, the tolerance mechanism is regarded as a significant advancement in the field of networks. Failures can occur for a variety of reasons. Transmission using multiple paths and/or multi-hops, on the other hand, impose additional energy consumption, lowering the life-time of the sensor network. Multicast transmission has been identified as a possible solution to such problems as delay and power exhaustion. Multicast routing can be an efficient way to achieve reliable transmission while also saving time and energy. As a result, QoS provision for RPL-based LLN communication could be realised. Multicast routing is the process of sending the very same message to multiple receivers at the same time within the radio transmission range of the transmitter.

Scalability and fault tolerance: The requirement for fault-tolerance grows as the size of the sensor system grows. Sensor systems for urban monitoring, battlefield surveillance, and habitat monitoring are being developed.

References

[1] P. High “Gartner: Top 10 Strategic Technology Trends For 2014” online at http://www.forbes.com/sites/peterhigh/2013/10/14/gartner-top-10-strategic-technology-trends-for-2014/#

[2] K. Mani Chandy and Leslie Lamport. Distributed Snapshots: Determining Global States of Distributed Systems. ACM Trans. Comput. Syst., 3(1):63–75, 1985.

[3] Ten H. Lai and Tao H. Yang. On Distributed Snapshots. Information Processing Letters, 25(3):153–158, 1987.

[4] FriedemannMattern. Virtual Time and Global States of Distributed Systems. In Parallel and Distributed Algorithms, pages 215–226. North-Holland, 1988.

[5] Xavier Etchevers, Gwen Salau¨n, Fabienne Boyer, Thierry Coupaye, and Noel De Palma. Reliable Self-deployment of Distributed Cloud Applications. Softw., Pract. Exper., 47(1):3–20, 2017.

[6] P. H. Su, C. Shih, J. Y. Hsu, K. Lin, and Y. Wang. Decentralized Fault Tolerance Mechanism for Intelligent IoT/M2M Middleware. In 2014 IEEE World Forum on Internet of Things (WF-IoT), pages 45–50, 2014.

[7] Xiaoli Xu, Tao Chen, and Mamoru Minami. Intelligent Fault Prediction System based on Internet of Things. Computers & Mathematics with Applications, 64(5):833– 839, 2012. Advanced Technologies in Computer, Consumer and Control.

[8] Martin Gerdes, YohanesBaptistaDafferiantoTrinugroho, Mari Næss, and Rune Fensli. Security, Reliability and Usability of mHealth Environments. In Mobile Health, pages 1043–1066. Springer International Publishing, 2015.

[9] J. P. G. Sterbenz. Smart City and IoT Resilience, Survivability, and Disruption Tolerance: Challenges, Modelling, and a Survey of Research Opportunities. In 2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM), pages 1–6, 2017.

[10] N. Mohamed, J. Al-Jaroodi, and I. Jawhar. Towards Fault Tolerant Fog Computing for IoT-Based Smart City Applications. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pages 0752–0757, 2019.

[11] Sen Zhou, Kwei-Jay Lin, Jun Na, Ching-Chi Chuang, and Chi-Sheng Shih. Support- ing Service Adaptation in Fault Tolerant Internet of Things. In Proc. of SOCA ’15, pages 65–72. IEEE, 2015.

[12] ZbigniewZieliski, Jan Chudzikiewicz, and JanuszFurtak. An Approach to Inte- grating Security and Fault Tolerance Mechanisms into the Military IoT. Springer International Publishing, 2019.

[13] S´ebastienGuillet, Bruno Bouchard, and AbdenourBouzouane. Safe and Automatic Addition of Fault Tolerance for Smart Homes Dedicated to People with Disabili- ties. In Trends in Ambient Intelligent Systems, pages 87–116. Springer International Publishing, 2016.

[14] Tuan Nguyen Gia, Amir-Mohammad Rahmani, TomiWesterlund, Pasi Liljeberg, and HannuTenhunen. Fault Tolerant and Scalable IoT-Based Architecture for Health Monitoring. In IEEE SAS, pages 1–6. IEEE, 2015.

[15] MasoudSaeidaArdekani, Rayman Preet Singh, Nitin Agrawal, Douglas B. Terry, and Riza O. Suminto. Rivulet: A Fault-tolerant Platform for Smart-home Applications. In Proc. of Middleware’17, Middleware ’17, pages 41–54. ACM, 2017

[16] Marc L´eger. Fiabilit´e Des Reconfigurations Dynamiques dans les Architectures `a Composants. PhD thesis, EcoleNationaleSup´erieure des Mines de Paris, 2009.

[17] FabrizioMontesi and Janine Weber. Circuit Breakers, Discovery, and API Gateways in Microservices. ArXiv, 2016.

[18] Lorenzo Alvisi, Karan Bhatia, and Keith Marzullo. Causality Tracking in Causal Message-logging Protocols. Distrib. Comput., 15(1):1–15, 2002.

[19] Stefano Porcarelli, Marco Castaldi, Felicita Di Giandomenico, Andrea Bondavalli, and Paola Inverardi. A Framework for Reconfiguration-Based Fault-Tolerance in Distributed Systems. In Rog´erio de Lemos, Cristina Gacek, and Alexander Romanovsky, editors, Architecting Dependable Systems II, pages 167–190, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[20] Tushar Deepak Chandra and Sam Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. J. ACM, 43(2):225–267, March 1996.