

The Pathology of Failures in IoT Systems

Mário Melo[1(✉)] and Gibeon Aquino[2]

[1] Federal Institute of Rio Grande do Norte, Natal, Brazil
mario.melo@ifrn.edu.br
[2] Federal University of Rio Grande do Norte, Natal, Brazil
gibeon@dimap.ufrn.br

Abstract. The presence of faults is inevitable in the Internet of Things (IoT) systems. Dependability in these systems is challenging due to the increasing level of dynamicity, heterogeneity, and complexity. IoT connects anything, anytime, and everywhere, introducing a complex relationship of interdependence, generating an increase in the susceptibility of the propagation of failures. The purpose of this study is to propose a pathology of failure in IoT Systems, exploring and characterizing faults, errors, failures, and their effects. This study investigates and classifies the source of faults, defines a taxonomy of the types of faults prone to happen, and defines the failure propagation model. As a result, the pathology establishes a common reference for fault, errors, and failures to be used by researchers and practitioners to improve tools for fault detection, fault diagnosis, fault tolerance, and fault handling in IoT Systems. This paper also proposes a failure propagation model for IoT systems that identify different combinations, paths, and fault-failure propagation effects.

Keywords: IoT · Dependability · Pathology · Taxonomy · Fault type · Fault source · Failure propagation

1 Introduction

Internet of Things (IoT) has attracted attention by projecting the vision of a global network of physical objects, enabling connectivity anytime, anyplace, for anything and anyone [18]. Nevertheless, this technological evolution came with a price: failures can cause fatal accidents, undermining their trustworthiness in the public eye [30]. For this reason, the dependability of IoT applications and infrastructures has become crucial in many contexts, particularly in critical domains [7,32]. Therefore, among the comprehensive variety of challenges when designing and deploying IoT systems, like their complexity, heterogeneity of the elements involved, and interoperability between different application domains, dependability is an open issue [9,26,38].

Fault management plays an essential role in maintaining system regular operation and decreasing faults, errors, and failures, leading to increased dependability property [14]. IoT systems often have particular requirements regarding fault

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12957, pp. 437–452, 2021.
https://doi.org/10.1007/978-3-030-87013-3_33

managing schemes, including specific fault characteristics due to their heterogeneity and complexity [11,19]. Although significant attention has been given to dependability aspects in IoT systems, no comprehensive, consistent, generally adopted fault characterization explains its cause and effects [23,29,30,40].

The objective of the present study is to propose a pathology of failures in IoT systems providing standard definitions and characterizations to assist researchers and practitioners in developing appropriate fault management solutions. To this aim, we focused on identifying the cause of faults, what are fault and error manifestation types, and a failure propagation behavior. Thus, the causal relationship between faults, errors, and failures is extended to identify which source of faults, the types of errors that can be activated, and how failure propagation occurs in IoT systems.

The term "pathology of failures" involves the creation and manifestation mechanisms of faults, errors, and failures displaying the causal relationship between them [4]. This concept can be generically interpreted: An active fault causes an error, and by propagation, many errors can be activated before a failure occurs. However, in medical science, pathology is a field of study primarily concerning the cause, origin, and nature of diseases. Similarly, a dependence among two systems could result in a pathological process that can develop along "linkage or connection" among them characterized by the activation of faults, the propagation of errors, and the occurrence of failures [4]. Thus, this research aims to establish the characterization, origin, and behavior of faults, errors, and failures that compose the pathology of failures in IoT systems.

The main contribution of this paper is to propose a pathology of failures for IoT systems establishing the context for the steps of the causal relationship. This contribution is composed of the pathological steps, which characterize the fundamental of the chain of dependability threats. The fault origin step identifies the sources of faults in order to provide a general view of fault distribution in IoT systems. Therefore, the fault origin distribution classification is established considering components and levels of abstraction present in IoT systems. The examination of the type of faults step identifies the causes of error occurrence, types of fault activation. Aiming to characterizes and classifies these types of faults and errors, we propose an IoT fault taxonomy to be used as a reference by researchers, developers, integrators, and developers for fault management solutions. The failure propagation behavior step establishes a failure propagation model considering the relation between levels of abstraction, systems, and the environment existing in IoT ecosystems.

As defined in this paper, the pathology of failures is useful for researchers, developers, and IoT practitioners, establishing the origin of threats to IoT systems, allowing the design of solutions that contemplate means to tolerate possible failures. In addition, it discusses a classification indicating the types of faults, establishing a common understanding, and describes its nature. Finally, this pathology supports researchers and practitioners in designing systems considering the interdependence between layers, systems, and the environment in which it is inserted.

The remainder of the article is organized as follows: The pathology of failure and the pathological step to define it are present in Sect. 2. In Sect. 3, the first step of the pathological process is presented, classifying the source of faults. Section 4 discusses and characterizes the types of faults defining a taxonomy. The failure propagation model is presented in Sect. 5. Finally, in Sect. 6, we conclude the paper and discuss future works.

2 The Pathology of Failures

In any given system, there is a causal relationship among faults, errors, and failures. This relationship is materialized when the fault-failure chain is activated, which includes the creation and activation mechanisms of faults, errors, and failures. A generic interpretation is that error only occurs when a fault is activated, and by propagation, can generate several errors before a failure occurs. Thus, this propagation, and chain instantiation, can occur via interaction between components, elements, and systems. These manifestations and relationships between dependability impairments are called the pathology of failures.

To define how pathology is instantiated in an IoT system, it is necessary to identify and classify a set of steps. Figure 1 shows the pathology steps in an IoT system. The Fault source identification step classifies where are the sources of threats. In order to achieve dependable systems, one effective way is to identify the source of faults and mitigate them timely, aiming to prevent failure propagation from causing significant in an IoT system and its dependents. The fault/Error taxonomy step defines a classification for the types of faults. Thus, identifying and establishing a comprehensive body of knowledge of faults and errors prone to happen in IoT systems will help researchers and practitioners propose suitable fault management strategies to mitigate these threats. A taxonomy of faults and errors describing characteristics, nature, and exemplification of faults and errors is proposed to this aim.

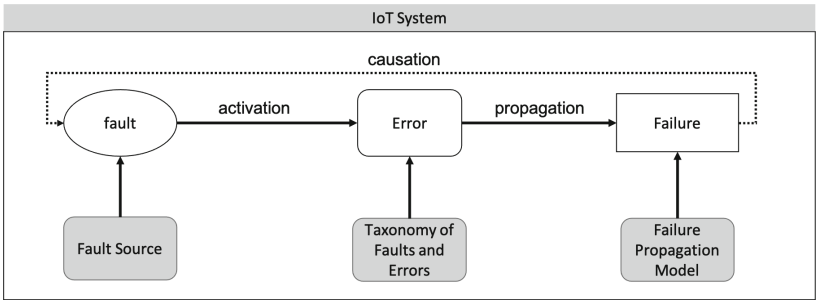

Fig. 1. Pathology of failures in IoT systems (adapted from [4])

IoT systems are composed of interrelated devices, which operate collaboratively and autonomously to deliver a service. If faults and errors were not handled

properly due to the interactions and dependencies between different devices in function or structure, propagation and cascading failures require understanding and investigation.

This paper considers an IoT system as a system composed of levels and components that deliver services to other systems. According to a given structure, the components of an IoT system can be physical or logical, which, in sequence, are composed by other systems recursively. Examples of physical components are sensors, actuators, gateway, cloud servers, routers, among others. Logically, the components can be embedded software systems, control and management software, and applications for end-users. At every level, all of these components act to deliver service of the level. From a more general perspective, all IoT System levels cooperate to deliver services to other external systems. There are different architecture models of IoT systems.We consider a four-level architecture to facilitate discussions of dependability and faults in this paper. The four generic levels include the perception level (e.g., sensors, actuators, and sensor networks), the communication or transport level (e.g., wired and wireless networks), the service or support level (e.g., cloud computing, storage area networks), and the application level (e.g., smart home, smart healthcare).

3 Sources of Faults

The first measure, usually necessary for a system to tolerate faults, is to detect them briefly. However, designers and developers should consider the causes of failures to assemble efficient fault tolerance strategies. Fault source classification is defined according to the system boundaries being internal and external [4]. These faults are related to the origin of the fault in relation to the system boundaries. Internal faults occur within the system boundaries and external faults outside system boundaries. However, this classification is too broad and does not go into the specifics and restrictions present in IoT systems. Therefore, understand and characterize these faults from an IoT systems viewpoint is essential to build effective dependability measures.

Aiming to identify fault sources and causes in IoT systems, we conduct an extensive analysis of the literature and state-of-the-art dependability related to IoT systems. As a result, we identified a comprehensive set of sources and causes through this investigation, which was subsequently organized into an IoT system levels viewpoint such as Perception Level, Communication Level, Service Level, and Application Level.

3.1 Perception Level

At this level, the causes of faults are directly related to obtaining and extracting the data associated with physical devices. The devices are typically different sensor nodes deployed to perform different measurements (e.g., temperature, humidity, ECG, EMG) and actuation. These sensor nodes are heterogeneous with diverse sensing and actuation mechanisms, processing, and communication.

Thus, this classification organizes the level in four major components, usually available in IoT devices, such as Sensors/Actuators, Processing Units, Network interfaces, Embedded Software. However, we know that some devices may not have any components or have other types of components not contemplated in this classification.

Sensors/Actuators. Physical components such as sensors, actuators, mobile devices, and mechanical components can produce faults. Some examples of faults are physical stress on components, inadequate enclosures, harsh environment deployment, and low energy could cause physical devices to malfunction. Furthermore, devices placed in hostile environments are vulnerable to faults related to disruption, direct intervention, or destruction. These types of faults are usually permanent, requiring manual intervention and replacement. Another fault type faced by physical objects is eletromagnetical interference. As the number of connected devices increases, radiation increases. This situation may influence measurements, transmitted messages, or control signals [37,39].

Faults that cause miss calibration may occur to the drift throughout the network lifetimes, which decreases the accuracy of sensor measurements. In addition, the calibration errors can produce specific faults such as offset faults (sensor measurements offset from the initial value by a constant quantity), gain faults (the rate of change of the measured data does not match with expectations over time), and drift faults (performance may drift away from the original calibration measurements) [21].

Processing Unit. This source of faults encompasses sensor nodes processing related components, such as memory, battery, and microprocessor. Faults on processing components could affect the quality or consistency of the stored data or their operations. For instance, memory bit flips or specific registers can corrupt the stored data. Limited memory fault or memory errors (memory leak, memory violation) can cause a system to crash or operate in degraded mode [1]. For energy supply, high battery temperature causes battery degradation leading to power outage. Likewise, battery low voltage may lead to data faults [21].

Network Communication. The form of communication utilized to connect devices can be wireless or wired, and each one can suffer different types of faults. For example, wireless communications are normally prone to interference (e.g., ambient noise, channel noise, multipath fading, radio frequency interference). Wired communication is vulnerable to a broken connector, material decay, and environmental effects on physical components (e.g., dirtying) [27,30].

Embedded Software. At sensor nodes, the software acts to processing and routing collected measurements. Thus, data processing faults can occur if data acquisition is not performed appropriately if the underlying sensors provide incorrect readings [36]. For example, an error in data aggregation can lead to

erroneous data to the upper levels. A typical data aggregation approach calculates the average correlated measured values of variables, sending only one message to the upper levels. If this level generates incorrect data, data aggregation results can lead to divergences from the real value. Software bugs are widespread, and some of them can generate cascading failures. Some generic examples are incorrect design specification, architectural faults, software design faults, incorrect algorithms [20].

3.2 Communication Level

The communication level is responsible for providing internetworking communication for the perception level. Specifically, the communication level transmits the sensed/aggregated data from the perception level to the upper levels through wired and wireless networks. The possibles causes of faults are a collision of messages, communication protocol violation, host environments, interference, dynamic changes in the network topology, and congestion. Therefore, we organize the faults in this level as Link and Connectivity.

Link. IoT communication links are established through wired and wireless communication mediums. Wired links have a constant rate of messages and tend to be more stable; however, there is exposition to disruption. On the other hand, wireless links are highly volatile and do not always yield the same message delivery rate [34]. Link instability can lead to faults in the routing paths [31]. Another fault related to both types of communication links is interference. If a communication link has heavy traffic, it could cause unavailability or message collision [34].

Moreover, devices can move to different locations where they become unreachable, resulting in complete data loss. For example, autonomous vehicles going to underground tunnels or places without cellular towers. Links also may fail when permanently or temporarily obstructed by an external object or environmental condition [27].

Connectivity. Connectivity faults could cause unacceptable delays and dropped or misguided messages [10]. Bugs at the connectivity routing network could cause incorrect destination message delivery, affecting several system paths and other collaborative systems. Routing is a fundamental feature for IoT systems [13,22]. Similar to the interference fault, the number of communicating devices might trigger connectivity faults, e.g., collisions or an overload of the network resulting in message collision. Another connectivity fault is protocol violation due to wrong message content due to different protocol versions or protocol mismatch.

3.3 Service Level

The service level is mainly qualified for storing and processing mass data in IoT systems [33]. At this level are cloud computing strategies, complex event

processing systems (CEP), and storage area networks (SAN). Faults originated from this level are linked mainly to hardware and software threats. Examples of hardware faults could be a hard disk that degrades its function due to temperature fluctuations, network connectivity issues, and service unavailability [25]. Software faults are related to contextual situations such as the wrong collection, erroneous inference, or the loss of events. Also, faults may be related to system control restrictions, such as input errors, deadline miss, and aging effects. Another probable cause is data fault, which involves threats due to data corruption, missing source data, and other flaws in the data [12]. Computation faults involve several types of hardware and infrastructure that could lead the system to fail due to storage access exceptions, server overload, network congestion, and software bugs [6].

3.4 Application Level

The faults related to this abstraction are related to application crashes due to bugs, operating system errors, unhandled exceptions, and incorrect usage [6]. However, the faults here have a higher amplitude than in the other abstractions since they can vary according to the applications' requirements. These faults are that most perceptible to the final users of an IoT system.

4 Taxonomy of Faults

Maintaining a dependable IoT system, even in the presence of faults, is a significant challenge. Therefore, understanding and characterizing these faults is vital for building solutions and strategies to attain dependability in these systems. Several aspects are significant for understanding and characterizing fault occurrence, particularly identifying fault types, origins, and probable causes. Thus, aiming to identify fault types in IoT systems, we conduct an extensive analysis of the literature and state of the art on faults, fault management, and dependability aspects in this domain. Through this analysis, we were able to identify a comprehensive set of classes and types, as presented in Fig. 2.

This proposed fault taxonomy has the purpose of being used as a reference by researchers, developers, system designers, and users, eliminating ambiguity and, eventually, enabling the use of a standard and comprehensible understanding of IoT faults. Furthermore, it will help system designers provide better fault management strategies to mitigate these faults to attain the dependability property.

This taxonomy organizes the faults by following the following classification: Duration, Location, Semantics, Behavior, Dimension, and Source Level (Detailed and Presented in Sect. 3). Although Azivienis et al. [4] and Raposo et al. [29] present and detail a taxonomy of faults, the first is to present a generic classification applicable to any system. The second extends this rating, including new views for the WSN domain, which is is part of the IoT domain. However, these presented fault taxonomy considers the few characteristics of WSN technologies, but their work provides insufficient information about current research trends, issues, and challenges, especially related to IoT systems faults.

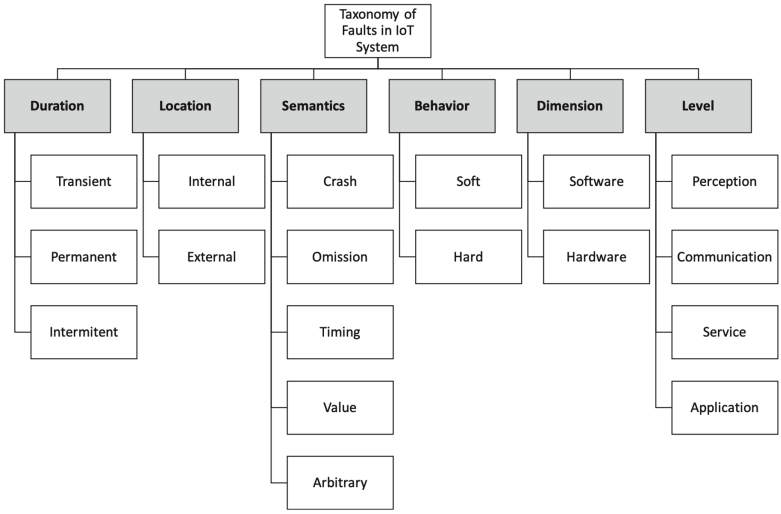

Fig. 2. Taxonomy of faults in IoT system

Duration. Another important type of fault is related to the duration of a fault. Hence, based on the period of the fault, faults can be further classified into persistent, permanent, and transient [4,17,21,23,24,29,30].

Transient. Transient faults are temporary faults that occur due to certain circumstances such as network congestion, changing climate conditions, among others. They occur for a short period and automatically rectify but may develop again. So, it is difficult to diagnose and manage transient faults. Some fault examples are radiation, input mistakes, security breaches.

Permanent Permanent faults that always produce errors. Some permanent faults examples are design faults, broken connectors, noise, system crashing, and aging effects.

Intermittent. Intermittent faults are developed in a longer period compared with transient faults. They may occur in an intermittent interval and have a specific frequency. Generally, intermittent faults cause behavior alteration in a system for some time duration and normal behavior at other times. This type of fault could occur in logical components (software) or physical components (hardware). Faults examples are effects of temperature on hardware, a transient fault like a short in the circuit activated by a specific input, reconfiguration faults, electromagnetic interference fault, incorrect computation fault.

4.1 Location

Each system has its boundary limit, delimiting the frontier between the system and the external environment. Consequently, faults can occur inside a system

or external systems (i.e., environment or other systems). Thus, faults can be classified based on the location which occurs. Regarding system limits, faults are classified as internal or external [4,23,29].

Internal. Internal faults may be originated from faulty physical components (e.g., broken device or connector) or introduced by design [30]. This type of fault generally affects a small number of components [24,29]. However, the lack of proper measures to rectify internal faults could cause fault propagation leading to external faults [23].

External. External faults originate from the environment (e.g., noise, radiation), other systems (e.g., system collaboration), or inputs (e.g., wrong or malicious usage of the system) [30]. Some causes of failure are channel noise, radiation, electromagnetic interference, operator mistakes, and environmental conditions [29]. External faults could propagate to internal systems generating internal faults. Furthermore, this type of fault may produce several types of errors and, consequently, system and component failures [4,29].

4.2 Semantics

This classification organizes the faults according to the behavior produced by the system when it is activated. Consequently, practitioners and researchers can understand the likely faults a system might exhibit to develop relevant fault tolerance strategies. They are as follows:

Crash. A crash fault occurs when an entity loses its internal state and cannot respond to the requests. Natural phenomena can cause this type of fault without human participation. Some reasons are battery depletion, message loss, power down, physical damage [2,8,10,21,28,40].

Omission. An omission fault is determined by a service eventually not responding to requests. Then, when a service fails to respond to incoming messages, caused by faults to send or receive messages in proper time. For instance, some reasons are radio interference, collision, channel overflow [8,10,21,28,40]. The malicious fault could also be considered an omission fault introduced by a human with a malicious objective or a natural fault [4].

Timing. A timing fault occurs when a service responds to a request out of the time interval. So, this fault happens when an entity response contains the correct value, but the response is received too late or too early. Timing failures will only occur if the application has timing constraints [2,8,10,21,28,40]. For example, even if the information is received correctly in real-time systems, it must be received within the maximum time defined in its restrictions. Thus, an overloaded sensor node that produces correct values with an excessive delay suffers from a timing failure.

Value. A fault value is produced when a service sends a timely response to a request with an inaccurate delivered value or an incorrect state transition.

For instance, a service performing aggregation of data generated by diverse sensor nodes could forward a result value to a gateway that does not correctly reflect the input data. Such situations could be caused by malfunctioning software, hardware, corrupt messages, or even security breaches generating incorrect data [2,21].

Arbitrary. Arbitrary faults include all previous types that cannot be classified in the described categories. This type of fault is also called Byzantine fault. In general, the cause of this fault may be a malicious service that behaves erroneously and fails to behave when interacting with other services and systems consistently. Some causes of this type of fault are security attacks, such as censorship, freeloading, misrouting, and data corruption [8,10,21,40].

4.3 Behavior

A fault occurrence could change the behavior of a system. In an IoT system, a fault could be classified according to system functioning after fault occurrence. So, based on a faulty IoT system's behavior, faults are classified as hard and soft [21,23,24,40].

Soft. When a system is subject to a soft fault continues operating with altered behavior. Some examples are offset faults that indicate a deviation in sensed data by an additive constant from the required data. Another fault type is the data loss fault produced when sensed data is missing from the time series for a given node.

Hard. Once it occurs causes general inactivity in the system. Some examples are depletion of the batteries leading to inaccurate data transmission or no transmission [34,35]. Other fault types are related to communication and hardware devices malfunction or breakage that require replacing faulty hardware.

4.4 Dimension

IoT system components can be simplified in two dimensions: first, they consist of hardware parts that physically support their operation and, second, they involve software/firmware modules that logically determine their functionality. Consequently, faults at any level can be classified in hardware or software [4,29].

Hardware. These are faults that originate in or affect IoT physical components and systems. Hardware faults examples are manufacturing imperfections, design errors, aging effects, disruption, environmental influence, electromagnetic interference.

Software. Classifies faults that directly or indirectly affect software components (i.e., programs or data). T This classification includes all faults generated in the development process and all software faults resulting from interaction with other systems. Examples of faults are software bugs, software aging, and security faults.

5 Failure Propagation

An IoT system could be defined as a large collection of independent and interconnected computing elements that cooperate to achieve a common goal [15]. These elements are organized in levels of abstraction, grouping elements with the same goal. If one of these elements could not handle its faults, another element inside its abstraction level must be handled, or a failure occurs. Avizienis et al. [4] define this behavior as a chain of threats that demonstrates a causal relationship between faults, errors, and failures. This chain could be, generically, interpreted as: when a fault is activated, it becomes a possible cause of an error. An error is a deviation of the system state that, through propagation, may lead to its subsequent failure. The propagation of failures in a system occurs if an abstraction level can not handle the failure, by itself or by a lower abstraction level, then propagated to upper abstraction levels.

IoT systems are an interrelated set of devices and entities that work collaboratively in obtaining data and processing that information. Thus, this interdependence could lead to a cascading failure when a propagated failure originating from a dependent component may have dynamic impacts, dependent on its occurrence time. Such sequence-dependent and dynamic effects pose a unique challenge [16]. Moreover, IoT systems are directly associated with the physical environment, so the context information affects system operation and environment. Therefore, failure propagation in IoT systems should consider the importance of context-awareness and ubiquity present in these systems and the impacts of failure propagation over them [3,28].

In order to model the propagation of failures in a system, we establish a causal relationship and dependence between the levels of abstraction and elements in IoT systems. We extend the cause-effect chain proposed by Bernardi et al. [5]] to an IoT System. Figure 3 exhibits the cause-effect relationship model for IoT systems. The diagram captures relationships between faults, errors, and failures according to its corresponding level of abstraction element. So a fault could affect an element (hardware, software, or communication unit), an error could affect a level of abstraction service performed by the faulty element or a faulty Level communication service, and, in general, a failure could affect the entire level of abstraction if errors are not recovered in due time. If an error reaches level boundaries, a failure is generated, affecting the service provided by the level.

Moreover, suppose an element service is affected by an error. In that case, this error can be propagated to another element service either delivered by the same element (internal propagation) or by another element communicating with the previous (external propagation). The Ecause-Eeffect association represents this error propagation.

In the context of IoT systems, a level of abstraction offers services that correspond to the delivered service by its element services or element communication service. So, faults are associated with the elements, while errors associated with the element service can potentially deliver. Thus, failure of a level of abstraction can be interpreted as an error not properly handle in the element communication service, causing an error propagation to the element service related to the

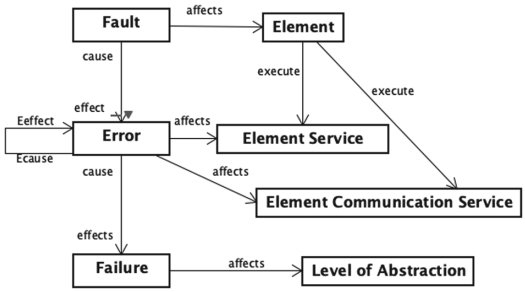

Fig. 3. IoT system cause-effect model

former or the failure of the faulty element, which is still unable to carry out any of its services, is perceived as an external fault by other interacting elements. In both cases, a failure may impact the entire level of abstraction, the system, and be observed as an external fault by other IoT systems interacting with the previous.

Although the cause-effect diagram illustrates the causal relationship between IoT elements, levels of abstraction, and threats, it lacks to describe a model for failure propagation considering systems and environment interaction. Considering this gap, we define a failure propagation model for IoT systems based on levels, systems, and environment interaction. Figure 4 depicts this model.

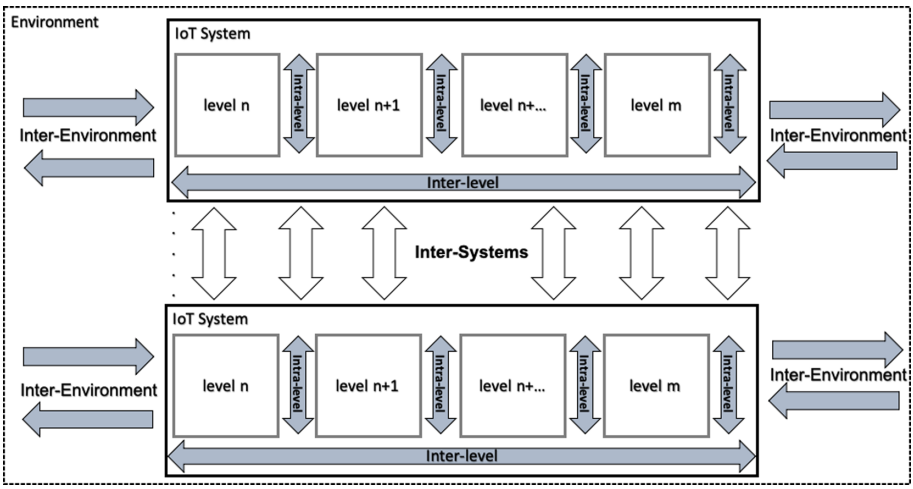

Fig. 4. Failure propagation model in IoT systems

IoT systems are formed by a set of abstraction levels providing services interacting with each other. However, each level has its own set of elements that interact with each other, providing and depending on services. Thus, a faulty

element can cause a succession of errors that, if not appropriately treated, could propagate internally into a certain level resulting in an intra-level propagation. However, if this failure crosses the abstraction level boundaries and reaches other levels, inter-level propagation occurs. These propagations could lead an entire system to fail. In the IoT domain, systems are inserted into an ecosystem that collaborates to provide common objectives. These linkages between systems could occasion the propagation of failures leading to inter-system propagation. This inter-system dependence reveals failure propagation into different types that eventually manifest as hazards due to the interaction with the physical world. Thus, it generates a failure propagation flow between the environment and IoT systems.

This model provides an IoT system viewpoint in which direction a failure propagation is probable to occur. Consequently, it provides a useful way of evaluating how errors propagate from one element to another due to their interactions. These interactions are complex due to the IoT system nature and increase with the number and type of interactions, resulting in emergent behaviors, including failure propagation, which is very difficult to predict. Thus, an important part of the system design is the analysis of failure propagation. This analysis increases the ability to project systems capable of limit the damage caused by any failure of their elements. Propagating a culture of evaluating scenarios of failures seems an essential part of improving dependability aspects.

To illustrate the applicability of the failure propagation model, we will use a smart home system as an example. This system includes several sensors and actuators such as temperature, presence, smart locks, video surveillance, and smart objects. All devices send the measurements to a gateway that collects and sends the data a cloud middleware. Each user can use a mobile app to monitor the house devices. A service provided by the perception level is that if a temperature sensor reaches a certain data threshold, the air conditioner is notified to decrease the house's temperature. If there is an incorrect computation fault in the gateway that calculates the house temperature, then an error occurs, and if the air conditioner remains off, a failure has occurred (intra-level propagation).

At the cloud level, a service provided is to unlock the using door using the mobile device. However, due to a bug fault introduced by a programmer, an error is raised and propagates and becomes a failure reaching the perception level by not open the required door (inter-level propagation). The communication internet communication is realized wirelessly through a set of antennas placed on the house's roof. However, the house is lashed with high-speed wind and rain, causing interference leading to data loss failing to obtain real-time data (inter-environment propagation). During a storm, a smart city disaster monitoring system sends incorrect information to the houses, informing only a short rain. However, an incorrect information fault has occurred, the houses should be warned to prepare for a massive hurricane, and due to this failure, people's lives are in danger (inter-system propagation).

6 Conclusion and Future Work

Faults in IoT systems, if they are not treated properly and become system failures, may interfere with user's everyday life or even lead to danger in their lives. However, to develop efficient fault management strategies, we define a pathology of failures for IoT systems. The pathology is divided into three steps: identifying fault sources, error/fault type definition, and failure propagation model.

The identification of fault source step classifies the fault origin classification for IoT systems organizing the faults at the level of probable occurrence. The defined levels are perception, communication, service, and application. On the error/fault type definition step, a taxonomy of faults was proposed. This taxonomy classifies the faults in the following classes: Duration, Location, Semantics, Behavior, Dimension, and Level. This taxonomy can be used by IoT researchers, developers, practitioners, and users. While presenting the taxonomy, examples of several faults and errors were given, and state-of-art references were provided. The failure propagation model step characterizes the fault-failure chain and the implications of failure propagation in an IoT system. Finally, the propagation model was established considering that an IoT system is a composition of levels of abstraction that interact with other systems and the environment. Thus, a failure propagates intra-level, inter-level, and inter-system and inter-environment failures.

In future research, we intend to propose a novel fault management framework for IoT Systems, which can manage fault-failures considering the level of interdependence and propagation.

References

1. Ali, A., Tixeuil, S.: Advanced faults patterns for WSN dependability benchmarking. In: Proceedings of the 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, pp. 39–48. ACM (2010)
2. Alrajei, N., Fu, H., Zhu, Y.: A survey on fault tolerance in wireless sensor networks. In: 2014 American Society For Engineering Education North Central Section Conference ASEE NCS Conference April, vol. 4 (2014)
3. Andrade, R.M.C., Carvalho, R.M., de Araújo, I.L., Oliveira, K.M., Maia, M.E.F.: What changes from ubiquitous computing to internet of things in interaction evaluation? In: Streitz, N., Markopoulos, P. (eds.) DAPI 2017. LNCS, vol. 10291, pp. 3–21. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58697-7_1
4. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secur. Comput. **1**(1), 11–33 (2004)
5. Bernardi, S., Donatelli, S., Dondossola, G.: Towards a methodological approach to specification and analysis of dependable automation systems. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS/FTRTFT -2004. LNCS, vol. 3253, pp. 36–51. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30206-3_5
6. Chetan, S., Ranganathan, A., Campbell, R.: Towards fault tolerance pervasive computing. IEEE Technol. Soc. Mag. **24**(1), 38–44 (2005). https://doi.org/10.1109/MTAS.2005.1407746

7. Ciccozzi, F., Crnkovic, I., Di Ruscio, D., Malavolta, I., Pelliccione, P., Spalazzese, R.: Model-driven engineering for mission-critical iot systems. IEEE Softw. **34**(1), 46–53 (2017)
8. Cristian, F.: Understanding fault-tolerant distributed systems. Commun. ACM **34**(2), 56–79 (1991)
9. Dar, K.S., Taherkordi, A., Eliassen, F.: Enhancing dependability of cloud-based IoT services through virtualization. In: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), pp. 106–116. IEEE (2016)
10. De Souza, L.M.S., Vogt, H., Beigl, M.: A survey on fault tolerance in wireless sensor networks. Interner Bericht. Fakultät für Informatik, Universität Karlsruhe (2007)
11. Denker, G., Dutt, N., Mehrotra, S., Stehr, M.-O., Talcott, C., Venkatasubramanian, N.: Resilient dependable cyber-physical systems: a middleware perspective. J. Internet Serv. Appl. **3**(1), 41–49 (2012). https://doi.org/10.1007/s13174-011-0057-4
12. Ganesh, A., Sandhya, M., Shankar, S.: A study on fault tolerance methods in cloud computing. In: 2014 IEEE International Advance Computing Conference (IACC), pp. 844–849. IEEE (2014)
13. Gia, T.N., Rahmani, A.M., Westerlund, T., Liljeberg, P., Tenhunen, H.: Fault tolerant and scalable IoT-based architecture for health monitoring. In: 2015 IEEE Sensors Applications Symposium (SAS), pp. 1–6. IEEE (2015)
14. Huangshui, H., Guihe, Q.: Fault management frameworks in wireless sensor networks. In: 2011 Fourth International Conference on Intelligent Computation Technology and Automation, vol. 2, pp. 1093–1096. IEEE (2011)
15. Iwanicki, K.: A distributed systems perspective on industrial IoT. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 1164–1170 (2018). https://doi.org/10.1109/ICDCS.2018.00116
16. Khan, F.I., Hameed, S.: Understanding security requirements and challenges in internet of things (IoTs): A review. arXiv preprint arXiv:1808.10529 (2018)
17. Khan, M.Z., Merabti, M., Askwith, B.: Design considerations for fault management in wireless sensor networks. In: 10th Annual Conference on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, UK (2009)
18. Kubler, S., Främling, K., Buda, A.: A standardized approach to deal with firewall and mobility policies in the iot. Pervasive Mob. Comput. **20**, 100–114 (2015)
19. Li, X., Ji, H., Li, Y.: Layered fault management scheme for end-to-end transmission in internet of things. Mob. Netw. Appl. **18**(2), 195–205 (2013)
20. Ma, R., Xing, L., Michel, H.E.: Fault-intrusion tolerant techniques in wireless sensor networks. In: 2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, pp. 85–94. IEEE (2006)
21. Mahapatro, A., Khilar, P.M.: Fault diagnosis in wireless sensor networks: a survey. IEEE Commun. Surv. Tutor. **15**(4), 2000–2026 (2013)
22. Misra, S., Gupta, A., Krishna, P.V., Agarwal, H., Obaidat, M.S.: An adaptive learning approach for fault-tolerant routing in internet of things. In: 2012 IEEE Wireless Communications and Networking Conference (WCNC), pp. 815–819. IEEE (2012)
23. Moridi, E., Haghparast, M., Hosseinzadeh, M., Jassbi, S.J.: Fault management frameworks in wireless sensor networks: a survey. Comput. Commun. **155**(March), 205–226 (2020). https://doi.org/10.1016/j.comcom.2020.03.011
24. Muhammed, T., Shaikh, R.A.: An analysis of fault detection strategies in wireless sensor networks. J. Netw. Comput. Appl. **78**, 267–287 (2017). https://doi.org/10.1016/j.jnca.2016.10.019. (November 2016)

25. Mukwevho, M.A., Celik, T.: Toward a smart cloud: a review of fault-tolerance methods in cloud systems. IEEE Trans. Serv. Comput. **14**(2), 589–605 (2018)
26. Ojie, E., Pereira, E.: Exploring dependability issues in IoT applications. In: Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, p. 123. ACM (2017)
27. Paradis, L., Han, Q.: A survey of fault management in wireless sensor networks. J. Netw. Syst. Manag. **15**(2), 171–190 (2007)
28. Power, A., Kotonya, G.: Complex patterns of failure: fault tolerance via complex event processing for IoT systems. In: Proceedings - 2019 IEEE International Congress on Cybermatics: 12th IEEE International Conference on Internet of Things, 15th IEEE International Conference on Green Computing and Communications, 12th IEEE International Conference on Cyber, Physical and So, pp. 986–993 (2019). https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00173
29. Raposo, D., Rodrigues, A., Silva, J.S., Boavida, F.: A taxonomy of faults for wireless sensor networks. J. Netw. Syst. Manag. **25**(3), 591–611 (2017)
30. Ratasich, D., Khalid, F., Geißler, F., Grosu, R., Shafique, M., Bartocci, E.: A roadmap toward the resilient internet of things for cyber-physical systems. IEEE Access **7**, 13260–13283 (2019). https://doi.org/10.1109/ACCESS.2019.2891969
31. Schmid, T., Dubois-Ferriere, H., Vetterli, M.: Sensorscope: experiences with a wireless building monitoring sensor network. In: Workshop on Real-World Wireless Sensor Networks (REALWSN 2005). No. CONF (2005)
32. Ştefan, V.K., Otto, P., Alexandrina, P.M.: Considerations regarding the dependability of internet of things. In: 2017 14th International Conference on Engineering of Modern Electric Systems (EMES), pp. 145–148. IEEE (2017)
33. Suo, H., Wan, J., Zou, C., Liu, J.: Security in the internet of things: a review. In: 2012 International Conference on Computer Science and Electronics Engineering, vol. 3, pp. 648–651. IEEE (2012)
34. Szewczyk, R., Polastre, J., Mainwaring, A., Culler, D.: Lessons from a sensor network expedition. In: Karl, H., Wolisz, A., Willig, A. (eds.) EWSN 2004. LNCS, vol. 2920, pp. 307–322. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24606-0_21
35. Tolle, G., et al.: A macroscope in the redwoods. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, pp. 51–63. ACM (2005)
36. Warriach, E.U., Aiello, M., Tei, K.: A machine learning approach for identifying and classifying faults in wireless sensor network. In: 2012 IEEE 15th International Conference on Computational Science and Engineering, pp. 618–625. IEEE (2012)
37. Xing, L., Li, H., Michel, H.E.: Fault-tolerance and reliability analysis for wireless sensor networks. Int. J. Perform. Eng. **5**(5), 419–431 (2009)
38. Xing, L., Zhao, G., Wang, Y., Mandava, L.: Competing failure analysis in iot systems with cascading functional dependence. In: 2018 Annual Reliability and Maintainability Symposium (RAMS), pp. 1–6. IEEE (2018)
39. Yaqoob, I., et al.: Internet of things architecture: recent advances, taxonomy, requirements, and open challenges. IEEE Wirel. Commun. **24**(3), 10–16 (2017)
40. Zhang, Z., Mehmood, A., Shu, L., Huo, Z., Zhang, Y., Mukherjee, M.: A survey on fault diagnosis in wireless sensor networks. IEEE Access **6**, 11349–11364 (2018). https://doi.org/10.1109/ACCESS.2018.2794519