

# Hybrid models for computing fault tolerance of IoT networks

**Bhupati Chokara, Sastry Kodanda Rama Jammalamadaka**

Department of Electronics and Computer Engineering, KELF University, Vaddeswaram, Guntur District, Andhra Pradesh, India

---

## Article Info

### Article history:

Received Dec 09, 2021

Revised Oct 26, 2022

Accepted Nov 12, 2022

---

### Keywords:

Complex structures

Fault tolerance

IoT networks

Networking topology

---

## ABSTRACT

Many Internet of Things (IoT) - based networks are being built to develop applications spanning multiple domains. Many small to large devices connected in various ways increases the risk of IoT networks failing. Small devices in the devices layer frequently fail due to their small size and high usage. Intermittent failures of the IoT networks lead to catastrophes at times. The IoT systems must be designed to be fault-tolerant. Fault tolerance of IoT networks must be computable so that the same can be considered while designing IoT networks. However, the computation of fault tolerance of IoT networks is complex, especially when heterogeneous structures are used for building a specific IoT network. Fault tree-based models are not suitable for computing fault-tolerance of complex models, which requires probability assessment. Hybrid fault tolerance computing models have been presented in this paper that consider both linear and probabilistic methods of computing the fault tolerance considering many complex networking topologies used in each layer of IoT networks. The fault-tolerance computing models are formal methods that can be used to compute the fault tolerance of any IoT network built with any internal processing. The accuracy of fault tolerance computing is 12.9% higher than other methods.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



---

## Corresponding Author:

Sastry Kodanda Rama Jammalamadaka

Department of Electronics and Computer Engineering, KELF University

Vaddeswaram, Guntur District, Andhra Pradesh, India

Email: drsastry@kluniversity.in

---

## 1. INTRODUCTION

Internet of Things (IoT) networks are built using different layers, with each layer built using different devices and networking topologies to connect those devices. The devices used in each layer differ in size and tolerance to different environmental variations on those devices [1]. Small devices often fail, not coping with adverse environmental conditions.

A correct model is required to calculate the fault tolerance of IoT networks when multiple topologies are used for networking with low-level devices connected through clustered devices. The devices situated in a layer need to communicate using different protocols. Some devices use strong protocols such as ethernet. Some devices use lightweight protocols such as message queue telemetry transport (MQTT), constrained application protocol (CoAP), and web sockets to meet certain purposes, ignoring failures, errors, flooding, and congestion.

The communication between lightweight and strong devices often fails due to huge latency [2], and the fault tolerance is normally low. The communication between the devices fails quite often due to insignificant paths affecting communications. The network used is generally hierarchical and star type, which localizes the faults and sometimes propagates up the hierarchy. The thin devices in the bottom-most layer of the IoT networks fail quite often as they get exposed to weather conditions and run out of power quite quickly. In IoT networks, the devices, the network, or the software that runs within devices can fail, and the faults can propagate down the line up to the root node.

There is a need to compute the fault rate of an IoT network considering different issues that include failures within the devices, network, and software. However, a device can be considered a failed device if any hardware-software failure occurs within those devices. In the absence of fault tolerance mechanisms built into IoT-based devices and networks, the networks are bound to fail and do not serve any purpose leading to huge wastage of effort and money.

IoT-based systems must be continuously available to be used effectively, fulfilling the application requirement. The IoT network may fail due to security breaches, malfunctions, or breakdowns. Some of the failures could be latency and sluggishness, leading to poor performance. IoT systems must be designed to incorporate fault tolerance mechanisms so that the IoT systems are highly available [3]. When a device or a network fails, the IoT networks become inoperative.

A computational model such as fault tree analysis (FTA) could be used to compute the fault tolerance of an IoT network if the network is simple and linear, without any redundancy or complicated networking. The linearity suffers, especially when clustering connects the devices in different layers. If the networking is complicated and the FTA methods cannot be employed, converting a complex to a linear structure is sometimes infeasible. Too much redundancy will cost the network heavily. A typical network provides alternate paths for communication so that redundant paths exist to affect the communication. Having more paths for effecting the same communication will also enhance the performance of the IoT networks. Incorporating redundancy built into the design of IoT networks to handle failures is one of the regularly built-in features. A different networking topology could connect the devices existing in a specific layer. The network topology is determined by the devices to be networked [4].

There is a need to compute the fault tolerance of the IoT networks considering different failure situations, whether due to hardware, software, or network. Complex networking topologies are to be used to create redundancy so that the failure situations can be handled without causing the breakdown. Fault tolerance of an IoT network must be computed to find the extent to which the network is tolerable to various faults happening within the IoT network to arrive at its reliability and availability for carrying out continuous operations.

FTA models cannot be used to compute the fault tolerance of a network built using complex networking topologies. Graph models are linear models which can be converted to FTA models. Most of the authors suggested various methods that contribute to improving the fault tolerance and not much covered the way the Fault rate or success rate of the entire network can be computed as a representable value.

Fault tolerance computing models must compute the entire network's fault tolerance, which involves linear and nonlinear models. Probability models are required for computing fault tolerance of the IoT networks when complex structures are used. Some IoT networks are built using both the linear and complex connectivity of the devices requiring the use of hybrid models that considers both FTA and Probability models for computing the fault tolerance of the IoT network.

The IoT networks are to be built by introducing redundancy at the device and network levels or some at the software levels to handle failures and keep the network running. Too much redundancy will cost the network heavily. Redundancy improves the fault tolerance of IoT networks, but it comes at a cost. A typical network provides alternate paths for communication, so the absence does not affect the network's overall performance [5].

The IoT network may fail due to security breaches, malfunctions, or breakdowns. When a device or a network fails, the IoT networks become inoperative. An IoT network is fault-tolerant when it continues to work even when a fault occurs. Sensors, for example, are fragile and therefore must be protected. Small device failures can lead to catastrophic failures in medical equipment. Even minor flaws in the equipment must be dealt with severely [6]. Data is lost when a device or network malfunctions. The networks must have mechanisms to preserve data even in case of failure. Non-volatile memory is used to store data even when the system fails. The IoT network must be fault-tolerant, even at increased costs.

Most failure situations can be handled by creating redundancy of devices, network paths, and software instances. All three elements' combined effect can be achieved through composite networks in different layers and then establishing an interconnection between the layers through appropriate means. It is difficult and straightforward to compute the fault rate when such composite networks are introduced into the IoT. Modeling a scenario where the adversary is malicious should allow for a dynamic topology in which system changes may occur without the (nonfaulty) processors being aware of these [7].

The fault tolerance of a communication system is calculated using several models. In situations where multiple faults occur simultaneously, several models must be developed. Authorized communication in large-scale open systems using directed and gate (AND) / or gate (OR) graphs has been proposed in the literature [8]. The FTA models help compute the fault tolerance of the linear IoT network. FTA models are not suitable when complex networking topologies are used to connect the things in the network. Lack of knowledge on the fault tolerance level of the IoT network will lead to deploying low available IoT networks to implement

applications that require high availability and response time. No single fault tolerance computing method will suffice for every possible IoT network built using complex structures that include different networking topologies, parallel and synchronous communication, and linear connectivity, represented by probabilistic and logical representations. Composite computing models are required for computing fault tolerance of any IoT network.

A directed multi-graph with colored edges is used by Burmester *et al.* [9], which is equivalent to a AND/OR graph to deal with independent faults. AND/OR graphs have been used to model problem-solving processes in artificial intelligence, and the same are used to model fault-tolerant computations with multiple inputs. An AND/OR graph is a directed graph with two types of vertices and labeled V-vertices. The graph must have one input (source) and one output (sink) vertex. The AND/OR graphs can also model the dependent faults.

General faults are modeled using adversary structures [10]. This structure is a monotone family of subsets of the system's components. Also, AND/OR graphs are used here. Adversary structures can represent general flaws. Graphs show the relationships between the devices. The fault tolerance of IoT networks is computed using the fault-tolerant and fog computing models [11]. The data transmission and energy minimization strategies, which also use tree-based graphs, have been used to deal with the issue of fault tolerance. A method to compute moments of failure times and a residual lifetime has been used based on continuous-time Markov chains for computing the fault rate of the devices connected to a network.

Fault tolerance schemes have been designed for modeling crash failures and silent data corruption within cloud computing systems using a parallel computing model [12]-[15]. The failure mechanisms have been modeled through the construction of an Optimisation problem. But developing such graphs is infeasible when an IoT network is complex.

Most of the approaches in the literature focus on increasing fault tolerance by using non-volatile memory to store data in case of failures and modeling crash failures and recommending provisions to counter such failures, using Markov models to predict uncertainty, and introducing redundancy. However, the overall fault rate of the entire network when such methods are implemented has not been presented. The computation of fault rate of the entire linear networks has been presented using the FTA models in the literature. No recommendations have been presented that focus on computing the fault rate when complex networking topologies are used to build the IoT network.

#### – Solution

To develop hybrid fault-tolerance computing models that can be used to compute fault tolerance of the IoT networks built using both linear and complex networking topologies. The hybrid models can be developed using linear FTA models and probability models that compute fault tolerance of complex topologies. The fault computation model computes the probability of failure and the probability model suited to a particular networking topology like cross bar, butterfly, and multistage needs to be used.

#### – Value proposition

The hybrid models are used by any industry that develops an IoT network to implement an application covering different domains. The industry concerned can compute the fault tolerance value of the network developed by them, make modifications without much adding to the cost, and deliver the same to the customers with the guaranteed operational time of the network. Major contributions of the paper include the following:

- 1) A method to convert simple clusters to linear models.
- 2) A method to convert complex structures to a linear model.
- 3) A probability-based computational model to compute the fault value of the IoT network built with complex topologies.
- 4) A method to compute the fault value of a linear IoT network.
- 5) A hybrid model to compute the fault value of an IoT network built with composite structures, including linear and complex topologies.

## 2. PROPOSED METHOD

In this, a hybrid approach to computing the fault rate of complex IoT networks is presented. The method follows the steps shown in Figure 1. The fault computation approach presented is based on the initial decision of whether the IoT network is linear or complex. If the model is linear, the clusters are converted into linear structures, the fault tree is constructed, the fault computation table is generated, and the overall fault is computed. Suppose the IoT model is found to be complex in that case, the complex structures are converted into butterfly models or through any topological model, and then the fault rate is computed using the probability models. The complex structures are then reduced into a single thing that is attached to the fault rate of the entire complex structure. The complex IoT model is then converted into a linear model for which a fault tree is constructed, a fault computing table is generated, and then the final fault rate of the entire network is computed.

If a cluster of mobile devices is used to inquire about the IoT, the mobile devices are formed into a cluster, and then the cluster is directly interfaced with the web services server forming into mobile edge computing as proposed by Samanta *et al.* [16]. Barrier scheduling is a frequently used technique to select a node that effectively communicates with the outside world. The barrier scheduling scheme considers the quality of service (QoS) considerations such as coverage, connectivity, and energy efficiency of the network. If there are too many clusters, each cluster is connected to a cluster head, which is one of the devices operating in the cluster. The failure of cluster nodes is traced, and the cluster is dynamically adjusted through weighted graphs as proposed by Thomas *et al.* [17]. The dynamically adjusted graphs are then treated as a linear model, and then FTA is constructed.

The controller layer is generally linear. A lightweight software-defined network (SDN) can be constructed using several microcontrollers that replace heavyweight controllers, as proposed by Chattopadhyay *et al.* [18]. The controller layer can then be converted into a cluster and a linear model. Reduced variable neighborhood search-based sensors data processing considering the reliability of data transmission and processing speed can be implemented in cluster head as proposed by Wang *et al.* [19]. The method proposed includes fault-tolerant data transmission, self-adaptive filtering, and data load reduction.

Kumar *et al.* [20] have presented a dynamic and artificial intelligent fault-tolerant mechanism for software-defined IoT, which considers network failures; the natural redundancy of functionality across devices, software, and hardware has been exploited by the method proposed by them. They have proposed to achieve fault tolerance by making the data backed up on virtual machines to maintain connectivity in case of network failures between the devices. Centralized edge computing within IoT puts overhead in cluster formation and management. Decentralized edge computing help achieve latency requirements by making the computing available closer to the computing edge of the users. Mudassar *et al.* [21] have proposed grouping heterogeneous edge nodes decentralized and processing the tasks in parallel to meet the deadlines. Fault tolerance has been the major criterion for the resources of limited edge devices that depend on the local information rather than the entire IoT network. The authors have proposed a method that runs in a decentralized manner to find the reliability of edge nodes locally while improving the overall network availability.

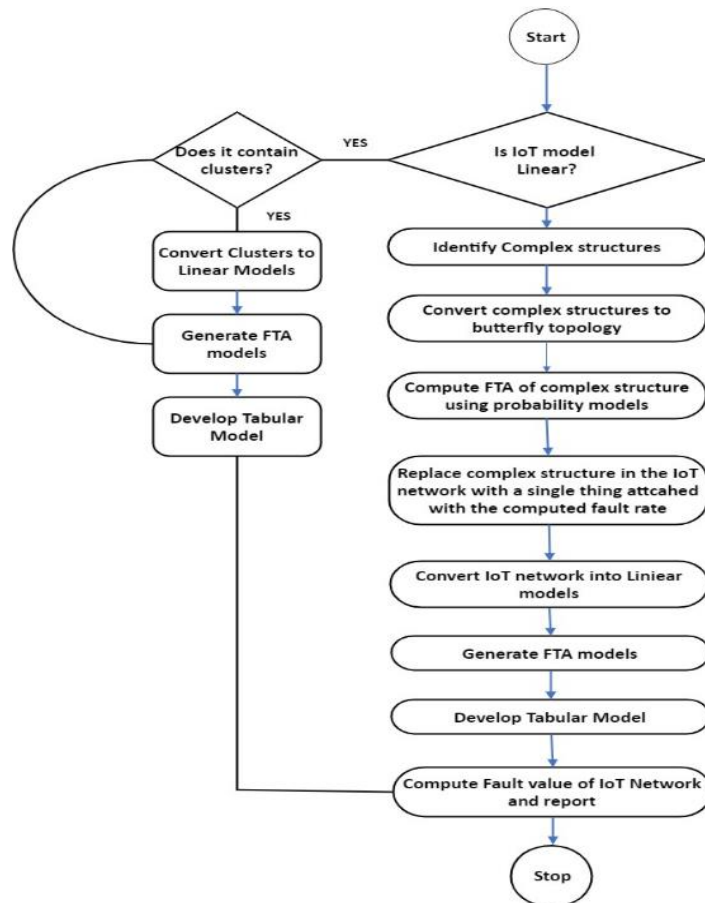


Figure 1. The proposed method of computing the fault tolerance rate of IoT networks using hybrid models

### 3. METHODS AND TECHNIQUES

#### 3.1. Computing the fault rate of linear IoT networks using the FTA method

IoT defect rate can be calculated FTA. Any network's fault tolerance can be calculated using FTA. The fault tree is built for a linear IoT network and then computes the fault. A fault tree depicts a system's flaws. Criticality, safety, operation, and environmental variables are shown. FTA shows the critical factors that can cause a network to fail. A fault tree is a graph that depicts the flaws that can cause a network to fail sequentially or concurrently. Software, hardware, and network defects can cause the failure of an IoT network.

Given a network, the fault tree of the network can be developed considering the failure model. The fault tree diagram also shows the relationships between the faults and how the faults propagate across the network leading to failures. The aggregate effect of all network faults can be attributed to the root node's fault rate. While fault tree analysis can predict severe issues, it cannot represent all faults that may occur over the life of a network.

The root node of the fault tree represents a networking event. Determining the root node's fault rate can be difficult when the system has many faults. As a result, fault tree analysis models a few essential and ancient faults. AND-OR gate logics connect devices to represent how the faults originated at a specific device and propagate to reach the network's root.

The FTA model connects devices and networks using AND-OR gates. Faults are propagated to the root node by the gates. FTA simulates higher-order events caused by lower-order events. Incoming errors affect outgoing devices hierarchically and sequentially. The OR gate sends the highest effective low order event to the output device, while the AND gate sends the combined effect of the incoming faults.

The devices are connected via AND-OR gates to form an FTA failure model. The failure rates of the incoming devices are determined using the AND-OR rules. AND-OR gates are not logic gates, so logic rules do not apply. Data related to the fault rates of the devices can be collected and stored in a database. The database also stores AND-OR connections between devices in different networking tiers. The database also stores each device's fault rate using mean time between failures (MTBF) data from the device makers. The required dependencies between devices are built with faults among linked devices in mind and their impact on outputs. Each layer is considered independently, and the FTAs formed for each layer are linked. The FTA diagram includes dummy devices to connect AND-OR gates wherever necessary.

#### 3.2. Converting device clusters into a fault tree

Device clusters exist in IoT networks in the device layer. Developing an FTA model is complicated when clusters are situated in the device layer of an IoT network. Figure 2 shows a device cluster involving three devices.

It is necessary to convert the device layer clusters into linear models to convert the network into a linear model. More intermittent devices are added, like the intermittent devices T12, T23, and T31 shown in Figure 3, to convert a cluster into a linear model. All clusters must be converted to linear models to compute the IoT network's fault tolerance level using the FTA method. FTA graph the linear models using AND-OR gates. Figure 4 depicts the linear model's fault tree diagram.

The conversion of the linear model to an FTA model is done using AND-OR logic, as shown in Figure 4. The conversion of clusters into linear models depends on the number of devices in a cluster and how the devices are connected in a specific layout. AND logics are used when an outgoing fault arises considering the combined effect of both the incoming faults. OR gate is used when one of the incoming faults has the dominating effect on the other fault. The fault rate of an outgoing device is the combined fault rate of the incoming devices if the AND logic is used. The fault rate of an outgoing device is the highest of the fault rate of the incoming faults when it comes to OR logic.

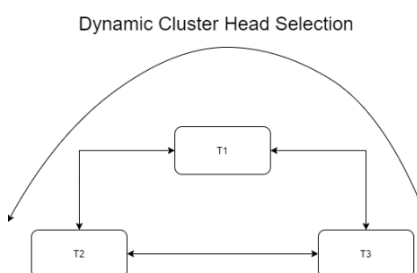


Figure 2. Cluster representation

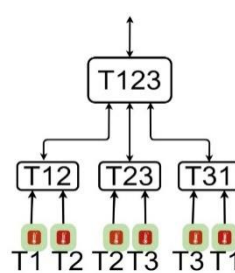


Figure 3. Converting a cluster into a linear model

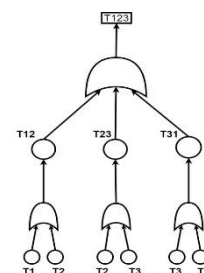


Figure 4. FTA equivalent for three-way cluster

### 3.3. Algorithm for generating FTA models from linear IoT networks

Given an IoT network containing device clusters, linear IoT models can be developed by converting clusters into linear models. The linear models relating to the clusters are combined with the remaining part of the network to form a wholesome linear network. The Algorithm that generates clusters in to a FTA model is shown in Table 1.

Table 1. Algorithm for generating clusters into linear models

Step number	Process undertaken
1	Capture an IoT network's hierarchy of hardware elements and update a database
2	Capture the clusters existing in the IoT diagram, convert it to a hierarchical model, and update the items in the database
3	Update the database with the failure rate of the devices obtained from the manufacturers
4	Capture the relationship (OR, AND) between each device and its predecessors and update the database
5	Generate the linear tree into a graph model

### 3.4. Algorithm for computing fault rate – computing the fault rate of linear IoT networks

Given a linear tree, the details of which are stored in a database, the fault rate of the network can be computed using the algorithm shown in Table 2 by implementing FTA-related rules. The fault rate of the respected devices used to develop the network are taken from the respective manufacturers. The incoming devices and the outgoing devices were all connected via gates. The AND-OR gates determine the fault rates affected by the precedence rule and device connectivity.

Table 2. Algorithm for computing the fault rate of a linear IoT network

Step number	Process undertaken
1	Query the elements from the database in the hierarchical order of preceding relationships connecting from the child nodes
2	Using AND-OR rules, compute the outgoing device's fault rate
3	To calculate an outgoing device's fault rate, multiply it by the incoming device's fault rate
4	If the relationship between the devices is an OR relationship, the outgoing device's fault rate is the lowest of the incoming devices' fault rates
5	Calculate the fault rate of the root device. A root device has no parents
6	Generate fault computation table

### 3.5. Handling complexity of device clusters from computing the fault rate of IoT networks

Handling clusters in IoT networks becomes difficult when multiple devices are linked together. The complexity can be reduced by converting a cluster into network topology and computing fault rates using probability models. Such conversions increase dramatically as the number of devices in a cluster grows. Butterfly, crossbar, mesh, and other topologies can be explored for networking a cluster.

Clusters are sometimes formed using complex structures, as shown in Figure 5. Developing linear models for such kind of cluster structure is quite complex. In such a case, different networking topologies such as butterfly and crossbar, can be used for converting complex cluster structures into networking topologies. A complex device cluster can be represented in network topologies such as butterfly and crossbar. Probability models can then compute the entire network's fault rate. The representation of the complex structure is shown in Figure 6 as a butterfly network.

The fault rate of a butterfly network can be computed using reliability models. A network with distinct topologies cannot have an FTA. The system's input goes through several stages to produce the output. The network uses multiple switches to process input and output. An alternate method for transporting data from the input to the output stage is chosen if the paths between stages are defective. Less than half of the failures of hierarchical networks occur in multistage networks.

$N \times N$  switches are needed when processing  $N$  inputs to produce  $N$  outputs. It is possible to achieve upper and lower broadcast, straight broadcast, and vertical broadcast. Two butterfly networks ( $2 \times 2$ ) must be used to build a  $4 \times 4$  network. The probability of a fault in a network is determined by the fact that at least one line out of a switch box at the output stage is functional, where  $ql$  is the probability that a link will fail, and  $\Phi(0)$  is the failure probability.

$$\Phi(0) = 1 - ql^2 \quad (1)$$

The following equation defines the probability that a switch box in stage  $i$  can fail, which is the failure of the entire network.

$$\Phi(i) = 1 - (1 - pl\Phi(i - 1))^2 \tag{2}$$

Sample calculations:

$$pl = 0.9, ql = 0.1, \text{ from the (1) and (2), } \Phi(i) = 1 - (1 - pl\Phi(i - 1))^2$$

$$\Phi(1) = 1 - (1 - 0.9\Phi(1 - 1))^2, \Phi(1) = 1 - (1 - 0.9\Phi(0))^2$$

$$\Phi(1) = 1 - (1 - 0.9(1 - ql^2))^2, \Phi(1) = 1 - (1 - 0.9(1 - (0.1)^2))^2$$

$$\Phi(1) = 1 - (1 - 0.9(1 - 0.01))^2, \Phi(1) = 1 - (1 - 0.891)^2, \Phi(1) = 0.98$$

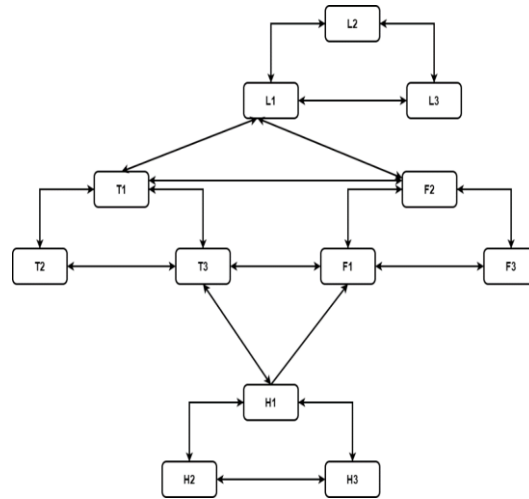


Figure 5. Complex cluster structure

**3.6. Representing a hybrid network built combining butterfly + linear network**

IoT layers can be built using alternative networking topologies. At the same time, some parts of the network are represented through a linear network; other parts of the network can be represented using complex topologies. The complex topologies can then be reduced into a single device in the network representing the entire topology. The fault rate of the device can be computed using its related probability model, which can be combined with the fault rate finding using FTA-related rules and regulations. A hybrid IoT network with a crossbar network for representing the device clusters and the rest of the network represented as a linear network is shown in Figure 7. The network representation is called a hybrid as both butterfly and linear networks are combined to form an overall network. The fault rate of the butterfly network is computed using its related probability model, and the fault rate of the rest of the network is computed using the FTA rules. The butterfly network is replaced by a single device to form a linear network. The fault rate of a single device replacing the butterfly network is obtained through computation achieved through the probability model.

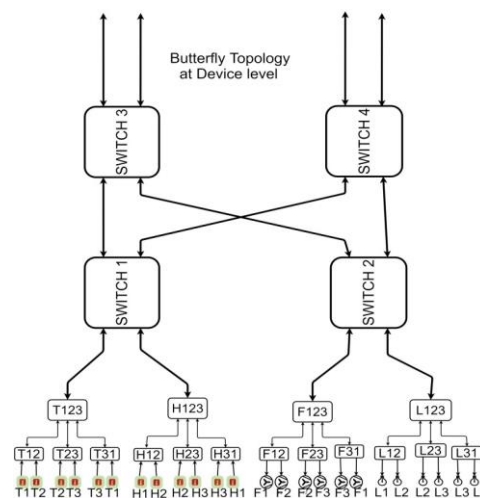


Figure 6. Conversion of a complex cluster into butterfly networking topology



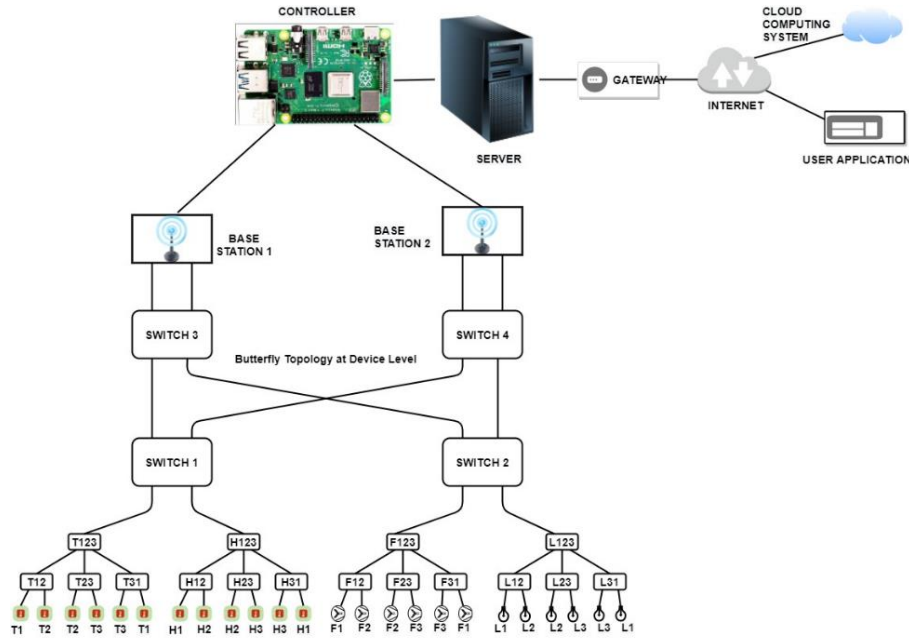


Figure 7. A hybrid IoT network (combining the butterfly with linear IoT structure)

The linear model that represents the hybrid model is shown in Figure 8. A single device replaces the butterfly part of the IoT network. The fault rate is the same as the fault rate of the butterfly network, which is computed using its related probability model.

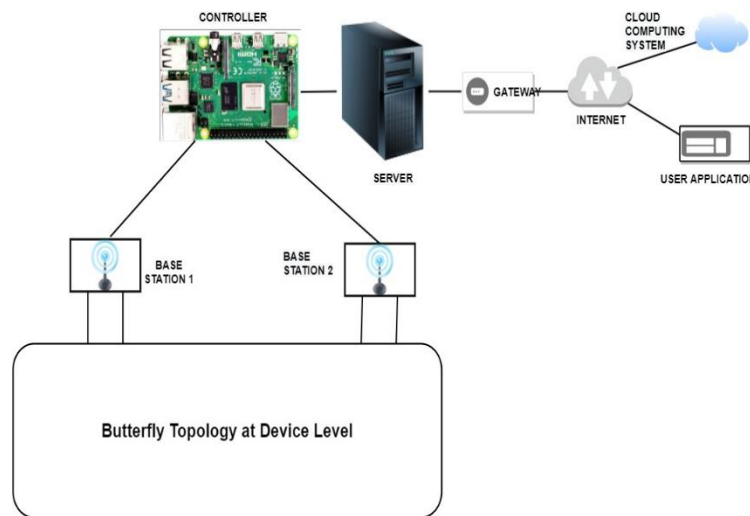


Figure 8. Equivalent linear model representing hybrid model

#### 4. RESULTS AND DISCUSSION

Accurate fault diagnosis and taking alternate actions are the key. Fault detection accuracy, false alarm rate, false positive rate, network lifetime, and throughput indicate the fault tolerance rate of the IoT network. However, no combined method exists that computes fault rate considering all the parameters. Fault tolerance rate is the parameter used to assess the fault tolerance of the IoT network.

Lavanya *et al.* [22] have used a support vector machine (SVM)-based learning model to diagnose the kind of fault with the wireless sensor network (WSN) network implemented at the device layer. Within the SVM model, they have used a grasshopper-based optimization method to determine the tuning parameters of the classifier. They have achieved a 99% accuracy in fault diagnosis, a 1.5% false alarm rate, a success rate of 0.710, and a life extension of 23 months when applied on a pilot project.



Energy-efficient fault detection and recovery management system represented in terms hidden poison Markov model has been proposed by Prasanth [23]. They have shown that fault diagnosis accuracy is 99%, and the false alarm rate is 2%. They have achieved a success rate of 0.700 and a life extension of 38 months when applied on a pilot project. An algorithm based on the multi-objective – deep reinforcement learning method has been proposed by Agarwal *et al.* [24]. They have achieved fault diagnosis accuracy of 98%, a success rate of 0.690, and a fault alarm rate of 3%. The lifetime of the IoT network is extended by 36 months.

An approach that helps select the cluster head and broker node simultaneously has been proposed by Bukhsh *et al.* [25]. They have implemented an energy-aware fault-tolerant system that schedules the messages for transmission within a broker node. They have achieved 98% accuracy in fault detection, a false alarm rate of 4%, The have achieved a fault rate of 0.698, and enhance the lifetime by 33 months

A sample IoT network is shown in Figure 9, which monitors temperature and humidity and controls air circulation through fans and air conditioning. The clusters are directly connected to the base stations for onward transmission to the controllers. Network and the linearized network are shown in Figure 9 and Figure 10. The FTA equivalent of the network shown in Figure 10 is shown in Figure 11. The sample fault computations generated are shown in Table 1.

The fault computation of the sample IoT network using the FTA method is carried out as per the FTA rules and regulations. The computations are shown in Table 3. The root node, or internet node, has a success rate of 0.717, which is the success rate of the entire IoT network. The network shown in Figure 10 is complex due to too many clusters. The complexity can be reduced by introducing a butterfly network replacing the clusters to make the network simple. The sample network that shows the butterfly network replacing the clusters is shown in Figure 12. Further simplification of the above network is undertaken by replacing the butterfly network with a single device attributed with a fault value that is the same as the fault value of the butterfly network computed using probability expressions.

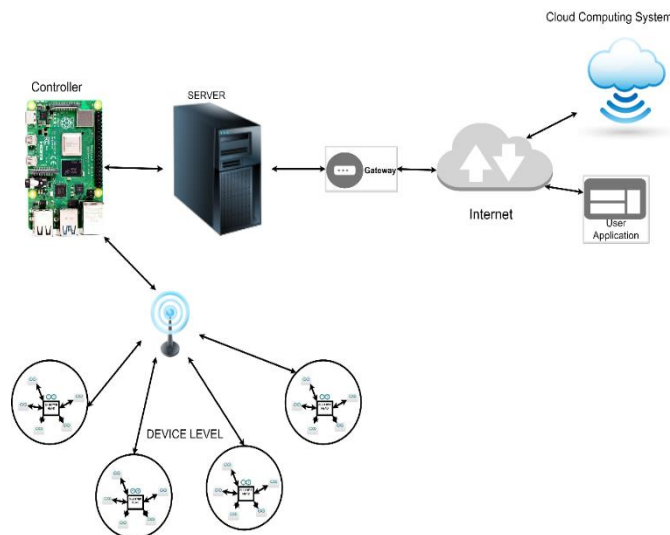


Figure 9. Sample IoT network with nonlinear cluster devices

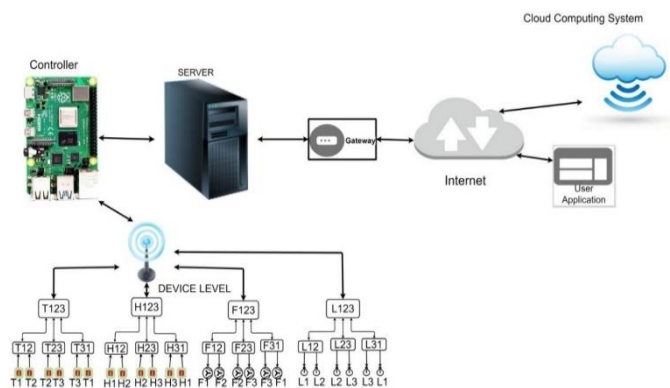


Figure 10. Linearized IoT network

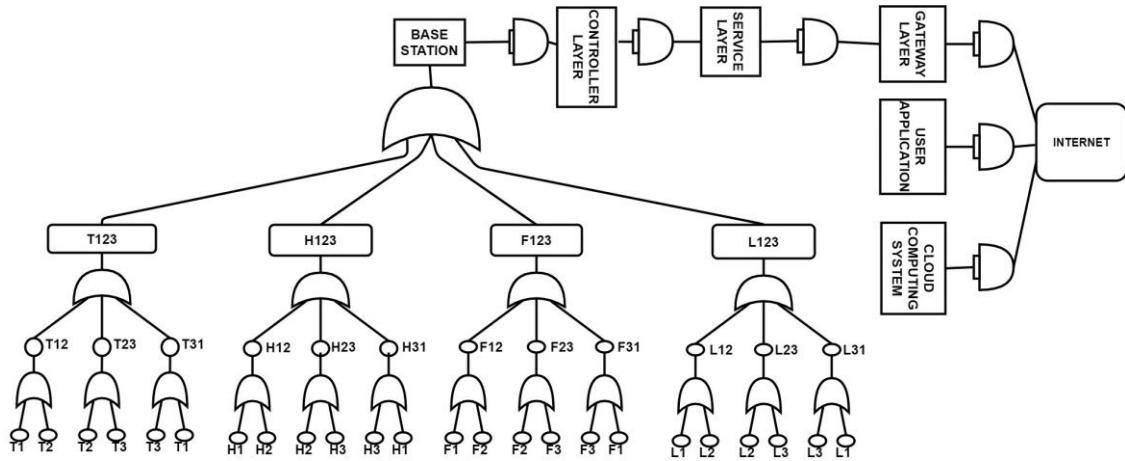


Figure 11. FTA diagram for sample linear IoT network

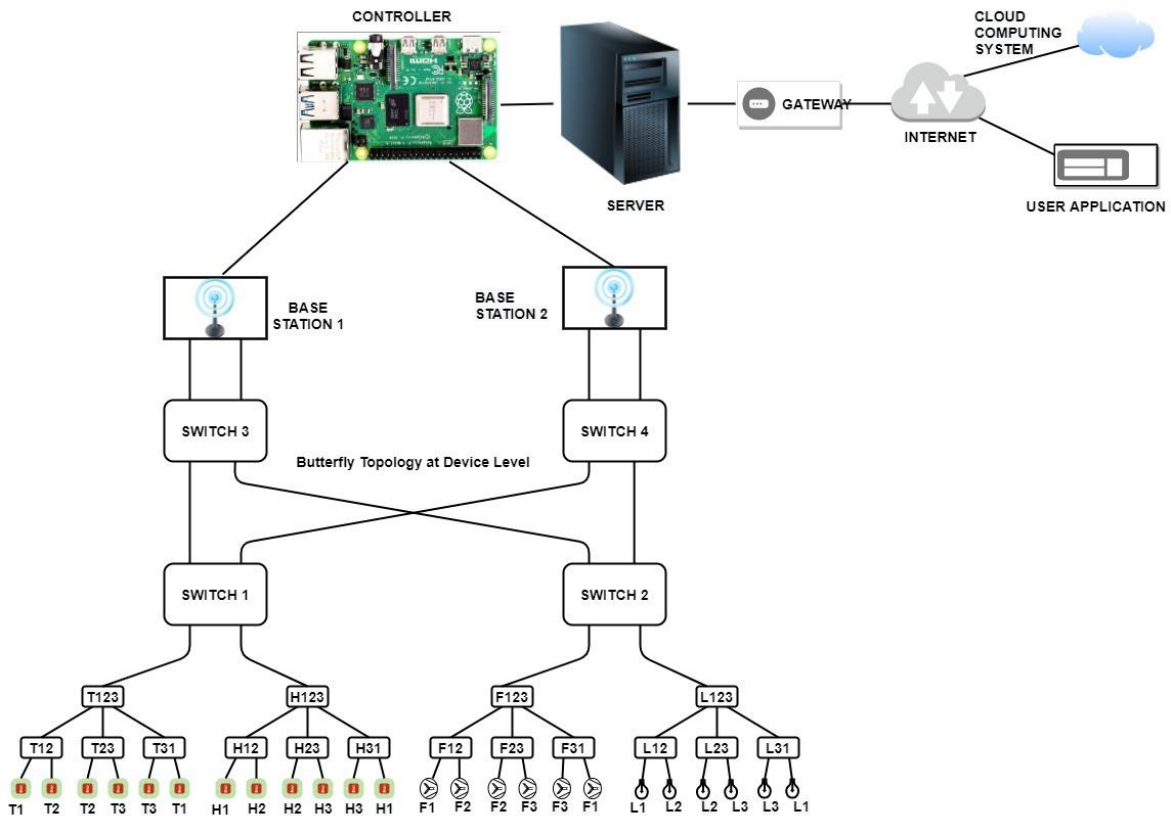


Figure 12. Sample hybridised IoT network built with butterfly network

The simplified network is shown in Figure 13. The FTA diagram is generated for the simplified network, shown in Figure 14, and the fault calculations of the same are shown in Table 4. It can be seen from Table 4 and Table 5 that there is an improvement in the success rate by 11.87% (0.796 – 0.710) when butterfly networks are used to replace the complex device clusters. The success rate is 0.796, and the false alarm rate is 0% as fault rates of manufacturer-supplied data are considered. The life of the IoT network is extended by 48 months due to using the cluster butterfly network at the device level. The model’s accuracy is 100%, as the computations are made on empirical formulations. The comparison of different models for estimating the accuracy of fault computation, compute fault rate, false alarm rate, and the extension of the life of the pilot project in months is shown in Table 5. From which it could be seen that the hybrid model of computing the success rate is the highest in the case of hybrid model.

Table 3. FTA calculation for sample linear IoT network

Serial number	Device	Success rate	Gates were used for the connection	Preceding devices					Combined Success rate
				Device name D1	Device name D2	Device name D3	Device name D4	Device name D5	
				Success rate S1	Success rate S2	Success rate S3	Success rate S4	Success rate S5	
1	Temp-sensor-1,2,3	0.950							0.950
4	T12, T23, T13-dummy	0.950	OR	T1	T2				0.950
7	T-123	0.950	OR	T12	T23	T13			0.950
14	H-123	0.950	OR	H12	H23	H31			0.950
15	FAN-1,2,3	0.950							0.950
21	F-123	0.950	OR	F12	F23	F13			0.950
22	Light 1,2,3	0.950							0.950
28	L-123	0.950	OR	L12	L23	L13			0.950
29	Base station	0.950	OR	T-123	H-123	F-123	L-123		0.950
30	Controller	0.900	AND	Base station					0.855
31	Restful server	0.900	AND	Controller					0.770
32	Gateway	0.980	AND	Restful server					0.755
33	Internet	0.950	AND	Gateway					0.717

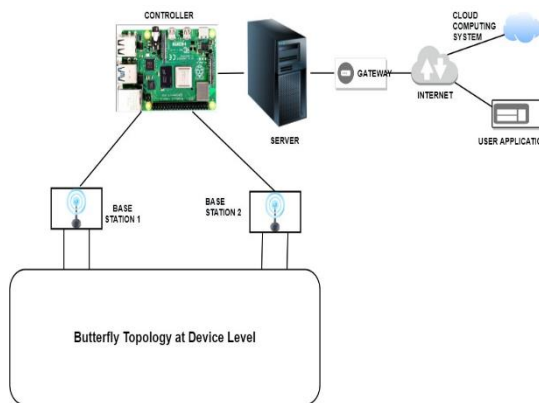


Figure 13. Simplified IoT network replacing the complex structures with a simple device

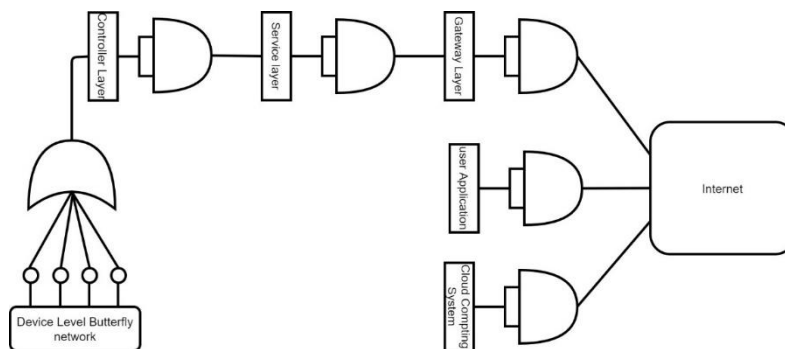


Figure 14. FTA diagram for simplified IoT network

Table 4. Fault calculations of the hybrid IoT network using FTA diagram

Serial number	Device	Success rate	Gates used for connection	Preceding Devices					Combined success rate
				Device name D1	Device name D2	Device name D3	Device name D4	Device name D5	
				Success rate S1	Success rate S2	Success rate S3	Success rate S4	Success rate S5	
1	Cluster head1	0.980							0.980
2	Cluster head2	0.980							0.980
3	Cluster head3	0.980							0.980
4	Cluster head4	0.980							0.980
5	Base station1	0.95	OR	Cluster head1 0.980	Cluster head2 0.980				0.980
6	Base station2	0.950	OR	Cluster head3 0.980	Cluster head4 0.980				0.980
7	Controller	0.900	OR	Base station1 0.950	Base station2 0.950				0.950
8	Services server	0.9	AND	Controller 0.950					0.855
9	Gateway	0.980	AND	Service server 0.855					0.838
10	Internet	0.950	AND	Gateway 0.838					0.796

Table 5. Comparative analysis of comparable models

Parameter	Hybrid model	SVM model [22]	Hidden Markov model [23]	Multi-objective deep hidden [24]	Cluster-broker selection [25]
Accuracy (percentage)	100.000	99.000	99.000	98.000	98.000
Success rate	0.796	0.710	0.700	0.690	0.698
False alarm rate (%)	0.000	0.150	0.200	0.300	0.400
Life extension (months)	48.000	23.000	38.000	36.000	33.000

## 5. CONCLUSIONS

The research focuses on developing linear and hybrid models that can be used to compute the fault rate of any IoT network. Simple clusters are to be converted into linear models, and complex clusters are to be converted into networking topologies, the fault rate of which can be computed using probability models. The success rate increased to 0.796 when butterfly models were used in the device layer from 0.717, achieved when linear transformations were carried out on the device clusters.

The accuracy of the fault rate is 100% when hybrid models are used for computing the fault tolerance of the IoT into which butterfly networks are introduced in the device layer, and the fault rate is computed using probability models. The success rate achieved using the hybrid model is 0.796, which is 12.36% more than the nearest model implemented through SVM. The life of the IoT networks gets extended by 48 months which is the highest compared to the nearest models. The false alarm rate is 0% which is the most significant achievement that one will get through the hybrid model.




## REFERENCES

- [1] N. Mardiyah, N. Setyawan, B. Retno, and Z. Has, "Active Fault Tolerance Control for Sensor Fault Problem in Wind Turbine Using SMO with LMI Approach," in *2018 5th International Conference on Electrical Engineering, Computer Science, and Informatics (EECSI)*, 2018, pp. 336-340, doi: 10.1109/EECSI.2018.8752721.
- [2] W. Najjar and J. -L. Gaudiot, "Network resilience: a measure of network fault tolerance," *IEEE Transactions on Computers*, vol. 39, no. 2, pp. 174-181, 1990, doi: 10.1109/12.45203.
- [3] M. T. Moghaddam and H. Muccini, "Fault-Tolerant IoT: A Systematic Mapping Study," *International Workshop on Software Engineering for Resilient Systems*, 2019, pp. 67-84, doi: 10.1007/978-3-030-30856-8\_5.
- [4] G. De Masi, "The impact of topology on Internet of Things: A multidisciplinary review," in *2018 Advances in Science and Engineering Technology International Conferences (ASET)*, 2018, pp. 1-6, doi: 10.1109/ICASET.2018.8376837.
- [5] A. Rullo, E. Serra, and J. Lobo, "Redundancy as a Measure of Fault-Tolerance for the Internet of Things: A Review," *Policy-Based Autonomic Data Governance, Lecture Notes in Computer Science book series (LNISA)*, vol. 11550, 2019, doi: 10.1007/978-3-030-17277-0\_11.
- [6] P. Vedavalli and C. Deepak, "Enhancing Reliability and Fault Tolerance in IoT," in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, 2020, pp. 1-6, doi: 10.1109/AISP48273.2020.9073174.




- [7] M. Burmester, Y. Desmedt, and Y. Wang, "A Critical Analysis of Models for Fault-Tolerant and Secure Computation," *Proc. Comm., Network, and Info. Security*, 2003, pp. 147-152. [Online]. Available: <https://www.cs.fsu.edu/~burmeste/bdw03.pdf>
- [8] M. Reiter and S. G. Stubblebine, "Towards acceptable metrics of authentication," in *Proc. 1997 IEEE Symposium on Security and Privacy*, 1997, pp. 10-20, doi: 10.1109/SECPR1.1997.601308.
- [9] M. Burmester, Y. Desmedt and Y. Wang, "Using Approximation hardness to achieve dependable computation," *International Workshop on Randomization and Approximation Techniques in Computer Science*, 1998, pp. 172-186, doi: 10.1007/3-540-49543-6\_15.
- [10] M. Hirt and U. Maurer, "Player Simulation and General Adversary Structures in Perfect Multiparty Computation," *Journal of Cryptology*, vol. 13, pp. 31-60, 2000, doi: 10.1007/s001459910003.
- [11] R. Oma, S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa, "A fault-tolerant tree-based fog computing model," *International Journal of Web and Grid Services*, vol. 15, no. 3, 2019, doi: 10.1504/IJWGS.2019.10022420.
- [12] X. Cui, Z. Hussain, T. Znati, and R. Melhem, "A systematic fault-tolerant computational model for both crash failures and silent data corruption," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2018, pp. 1-8, doi: 10.1109/ICIN.2018.8401596.
- [13] J. B. Dugan, S. J. Bavuso, and M. A. Boyd, "Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems," *IEEE Transactions on Reliability*, vol. 41, no. 3, 1992, doi: 10.1109/24.159800.
- [14] L. Vihman, M. Kruusmaa, and J. Raik, "Systematic Review of Fault-Tolerant Techniques in Underwater Sensor Networks," *Sensors*, vol. 21, no. 9, 2021, doi: 10.3390/s21093264.
- [15] Bhupathi and S. Jammalamadaka, "A framework for effecting fault tolerance within IoT network," *Journal of Advanced Research in Dynamical and Control Systems*, vol. 10, pp. 424-432, 2018. [Online]. Available: [https://www.researchgate.net/publication/325735593\\_A\\_framework\\_for\\_effecting\\_fault\\_tolerance\\_within\\_IoT\\_network](https://www.researchgate.net/publication/325735593_A_framework_for_effecting_fault_tolerance_within_IoT_network)
- [16] A. Samanta, F. Esposito, and T. G. Nguyen, "Fault-Tolerant Mechanism for Edge-Based IoT Networks with Demand Uncertainty," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16963-16971, 2021, doi: 10.1109/JIOT.2021.3075681.
- [17] D. Thomas, M. Orgun, M. Hitchens, R. Shankaran, S. C. Mukhopadhyay, and W. Ni, "A Graph-Based Fault-Tolerant Approach to Modeling QoS for IoT-Based Surveillance Applications," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3587-3604, 2021, doi: 10.1109/JIOT.2020.3022941.
- [18] S. Chattopadhyay, S. Chatterjee, S. Nandi, and S. Chakraborty, "Aloe: Fault-Tolerant Network Management and Orchestration Framework for IoT Applications," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2396-2409, 2020, doi: 10.1109/TNSM.2020.3008426.
- [19] K. Wang, Y. Shao, L. Xie, J. Wu, and S. Guo, "Adaptive and Fault-Tolerant Data Processing in Healthcare IoT Based on Fog Computing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 263-273, 2020, doi: 10.1109/TNSE.2018.2859307.
- [20] S. Kumar, P. Ranjan, P. Singh, and M. R. Tripathy, "Design and Implementation of Fault Tolerance Technique for Internet of Things (IoT)," *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2020, pp. 154-159, doi: 10.1109/CICN49253.2020.9242553.
- [21] M. Mudassar, Y. Zhai, L. Liao, and J. Shen, "A Decentralized Latency-Aware Task Allocation and Group Formation Approach with Fault Tolerance for IoT Applications," *IEEE Access*, vol. 8, pp. 49212-49223, 2020, doi: 10.1109/ACCESS.2020.2979939.
- [22] S. Lavanya, A. Prasanth, S. Jayachitra, and A. Shenbagarajan, "A Tuned classification approach for efficient heterogeneous fault diagnosis in IoT-enabled WSN applications," *Measurement*, vol. 183, 2021, doi: 10.1016/j.measurement.2021.109771.
- [23] A. Prasanth, "Certain Investigations on Energy-Efficient Fault Detection and Recovery Management in Underwater Wireless Sensor Networks," *Journal of Circuits, Systems and Computers*, vol. 30, no. 08, 2021, doi: 10.1142/s0218126621501371.
- [24] V. Agarwal, S. Tapaswi and P. Chanak, "Intelligent Fault-Tolerance Data Routing Scheme for IoT-enabled WSNs," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16332-16342, 2022, doi: 10.1109/JIOT.2022.3151501.
- [25] M. Bukhsh, S. Abdullah, A. Rahman, M. N. Asghar, H. Arshad and A. Alabdulatif, "An Energy-Aware, Highly Available, and Fault-Tolerant Method for Reliable IoT Systems," *IEEE Access*, vol. 9, pp. 145363-145381, 2021, doi: 10.1109/ACCESS.2021.3121033.

## BIOGRAPHIES OF AUTHORS



**Bhupati Chokara**    is presently serving as an Assistant professor of Electronics and Computer Engineering at KLEF University. He is also a scholar at the same University and about to graduate from his Ph.D. program. He has been actively managing a huge IoT infrastructure at KLEF University. He has pioneered many aspects that enhance the fault tolerance of many mission-critical IoT-based applications. He has published two SCI papers and 4 SCOPUS Indexed papers. He can be contacted at email: [bhupati@kluniversity.in](mailto:bhupati@kluniversity.in) and [bhupati406@gmail.com](mailto:bhupati406@gmail.com).



**Sastry Kodanda Rama Jammalamadaka**    has 30 Years of IT Experience and 16 Years of academic experience. Double doctorate in CSE and MGT. Has been qualified in BE (Electrical), ME (Control Engineering), MBA (Finance), and M.Sc. (Stat). Has guided 23 Scholars and has published 288 Papers, 21 In SCI, 124 in SCOPUS, and the rest indexed in Google Scholar. The number of citations of the papers published by Dr. Sastry has crossed 800. He specializes in AI, ML, DL, Embedded Systems, IoT, Web technologies, Cloud computing, Software engineering, and Big Data Analytics. Twenty-three scholars have been awarded Ph.D. under the guidance of Dr. Sastry. He is presently working as a Professor at KLEF University, Vaddeswaram, Guntur District, India. He can be contacted at email: [drsastry@kluniversity.in](mailto:drsastry@kluniversity.in) and [drjksastry@gmail.com](mailto:drjksastry@gmail.com).