DESIGN A FAULT TOLERANT DATA COLLECTION NETWORK STRUCTURE IN IoT

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Anirudh Pal Dev 191259
Anshul Thakur 191436

Under the supervision of

Mr. Arvind Kumar
to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology
Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Design a Fault Tolerant Data Collection Network Structure in IoT"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **Mr. Arvind Kumar** Assistant Professor (Grade-II), Department of Computer Science and Engineering.
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

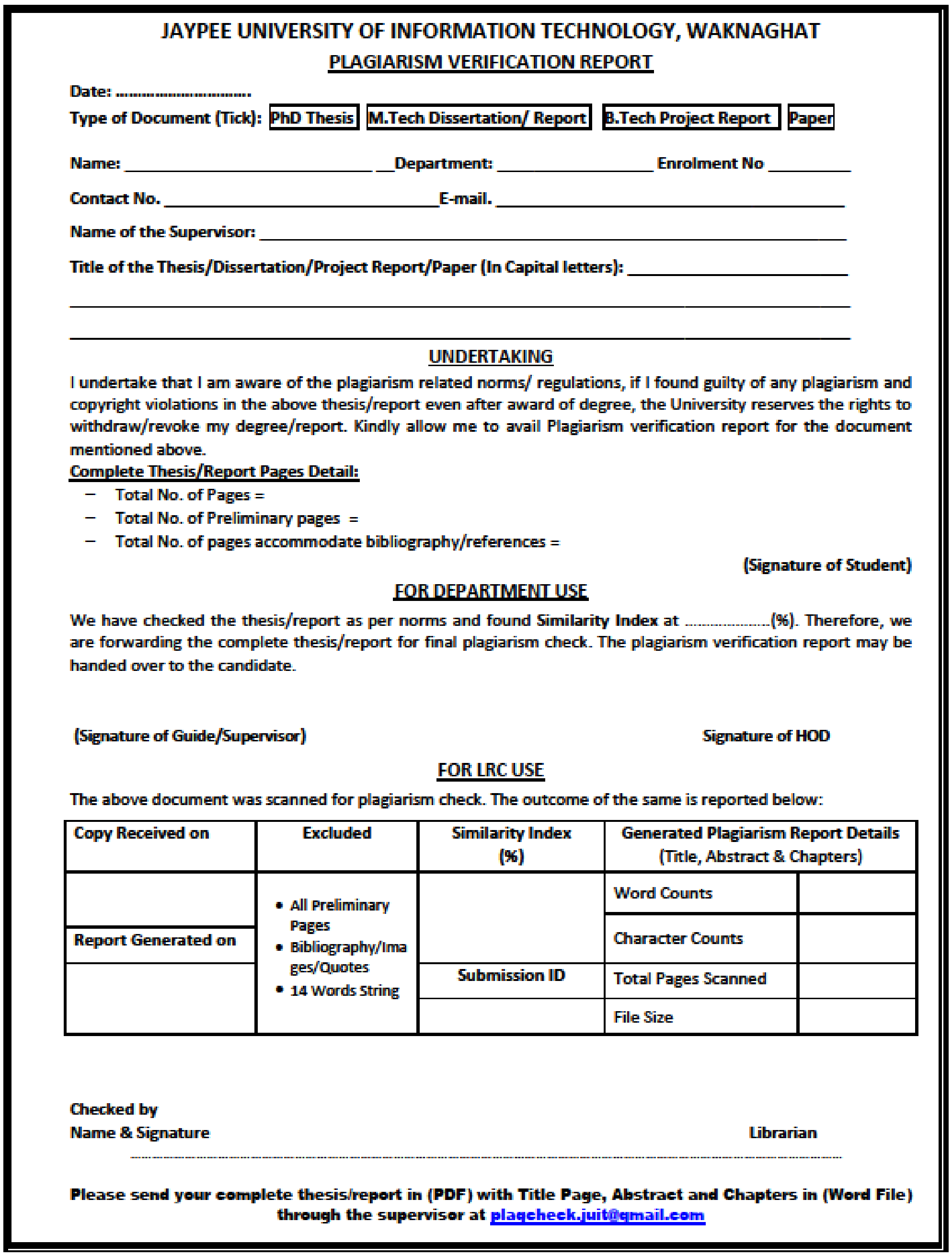
(Student Signature)
Anirudh Pal Dev, 191259

Anshul Thakur, 191436

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Supervisor Name - **Mr. Arvind Kumar**
Designation - Assistant Professor (Grade-II)
Department name - Computer Science and Engineering
Dated: 03/05/2023

Plagiarism Certificate



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ____________________ Department: ______________ Enrolment No ________

Contact No. ______________________ E-mail. ______________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

– Total No. of Pages =
– Total No. of Preliminary pages =
– Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature Librarian

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

First, I express my heartiest thanks and gratefulness to Almighty God for His divine blessing to make it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Mr. Arvind Kumar, Assistant Professor (Grade-II),** Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of "**Cloud Computing**" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Mr. Arvind Kumar**, Department of CSE, for his kind help in finishing my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educated and non- instructed, who have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

(Student Signature)

Project Group No.: 124

Student Name: Anirudh Pal Dev, Anshul Thakur

Roll No.: 191259, 191436

TABLE OF CONTENT

LIST OF ABBREVIATIONS

Sr. No.	**List**	**Full Form**
1	WSN	Wireless Sensor Networks
2	IoT	Internet of Things
3	BS	Base Stations
4	CTP	Collection Tree Protocol
5	DADCNS	Delay Aware Data Collection Network Structure
6	CDCT	Concurrent Data Collection Tree
7	ETX	Expected Transmission
8	TDMA	Time Division Multiple Access

LIST OF FIGURES

ABSTRACT

The Internet of Things (IoT) is a revolutionary invention that ushers in the digital era in the physical world. The IoT boom has resulted in an explosion of new products on the market, including voice assistants, smart cars, and smartphones. IoT technology exists to enhance and simplify daily conveniences. The Internet of Things (IoT) is a network of physical objects that are represented digitally. It can be compared to a system of embedded, internet-connected gadgets with sensors that collect data on the objects they are embedded in. The Internet of Things is bringing about a new wave of automation, intelligence, and control. IoT sensors monitor the environment for changes or anomalies and provide feedback to people or systems so they can react appropriately. Real inclusion and exclusion criteria, as well as a thorough analysis, were used to choose the primary analysis. IoT infrastructures often connect sensors, controllers, and the outside world through a central node or gateway.

The adage "Necessity is the Mother of Invention" sums up the creation of WSN nicely. As technology has improved, systems have accelerated the creation of smart sensors. As a result, a network can be built using different sensor nodes. Applications for wireless sensor networks include military operations, monitoring vital infrastructure, and data collection in dangerous locations. Weather that poses a threat has an influence on the WSNs that remote sensor organizations use to monitor their systems. Sensor hubs are designed to function independently under potentially hazardous circumstances. A sensor node's lifespan might vary from a few hours to months or years, depending on the environment it functions in. As a result, WSNs are susceptible to defects, which are likely to occur frequently and randomly. The ability to distinguish between malfunctioning and functioning nodes due to errors is crucial in a sensor network. Fault management poses a substantial design difficulty for WSNs because to all of these factors. Faults must be treated with particular care and respect. As a result, WSNs are susceptible to flaws, which are unpredictable and likely to occur frequently. There will always be flaws in a sensor network, thus it's critical to distinguish between malfunctioning and functional nodes. Fault management poses a substantial design difficulty for WSNs because to all of these factors. Faults must be treated with particular care and respect. As a result, WSNs are susceptible to flaws, which are unpredictable and likely to occur frequently. There will always be flaws in a sensor network, thus it's critical to distinguish between malfunctioning and functional nodes. Fault management poses a substantial design difficulty for WSNs because to all of these factors, faults must be treated with particular care.

The project's goal is to create a network structure for a fault-tolerant data collection system that can handle various faults and data transmission failures. It offers a trustworthy, fault-tolerant framework that offers a dependable and fault-tolerant structure that may be used in future IoT applications on different patterns.

Keywords: Fault-Tolerance, Redundancy, Internet of Things, Dynamic Reconfiguration, Remote Monitoring, Scalability, Wireless Sensor Networks (WSNs), Sensors Management, Network Architecture, Network Topology, Data Transmissions, IoT Applications.

Chapter-1

INTRODUCTION

1.1 Introduction

70% of the world's inhabitants are projected to live in urban regions by 2050. Cities must make use of contemporary information and communications technologies to promptly deliver up-to-date information to their residents in order to facilitate such rapid expansion. The Internet of Things (IoT) is one of the technological advancements that is frequently cited as a viable solution. Many of the current smart cities have standalone, non-interoperable IoT systems. Instead of creating many discrete closed form systems, IoT devices mounted on various assets should be networked to increase efficiency and realize the full potential of smart cities. Both public and private IoT infrastructures should be shared to prevent overprovisioning and pointless redundancy. For example, a smart city typically has a number of Data Things. For instance, a smart city usually includes a range of applications; hence, different apps may be compliant with the same set of devices. Additionally, it's possible that numerous programmers will use the same information. The reduced deployment costs, time, and maintenance needs of the shared infrastructure in such a case will benefit applications greatly. Additionally, using the infrastructure-as-a-service business model, identical infrastructure can be made available to apps for a fee. The allocation of resources, data collecting, data privacy and security, energy use, and a few other issues must all be resolved. Modern protocols for IoT and WSN applications have been primarily developed to enhance network hierarchy and performance. For instance, the changes investigated put a strong emphasis on automating job management and maintenance as well as boosting resilience against failures (such as electrical or communication failures). As a result, third-party management protocols (like SNMP) keep track of the node's state and issue alerts. The majority of Internet of Things (IoT) applications are made up of software components that are dispersed throughout the network as perception, data processing, storage.

The network shown in Figure [1] shows the majority of IoT networks concentrate on fault-tolerant data analysis and transmission, an architecture-based characteristic of the ensuing data processing and storage modelling are defined-

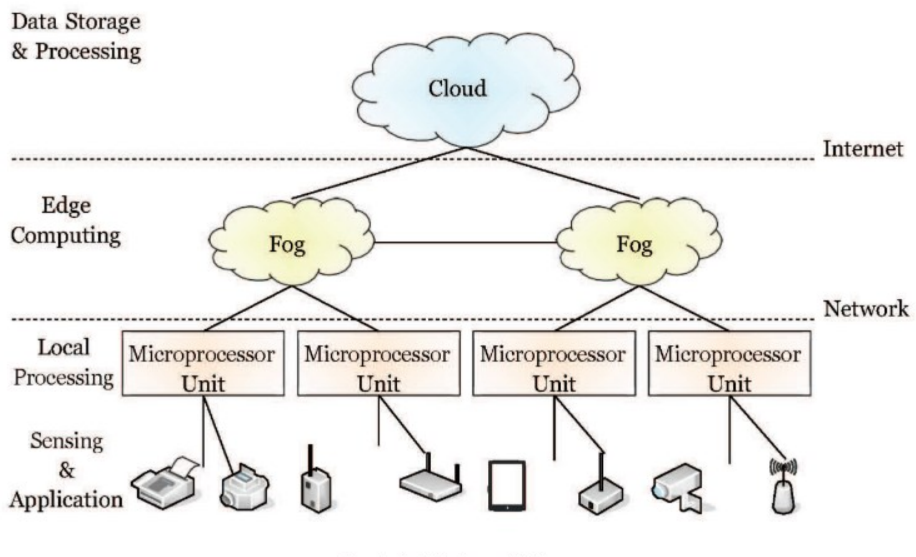


Fig. 1: Architecture of IoT

Fig. 1.1: Architecture of Internet of Things (IoT)

Since multiple applications may query data sources simultaneously and the freshness of the data must be maintained, the process of data collection presents a significant challenge. Depending on this factor, either a single node or several nodes dispersed across the IoT system should run data analysis software. In other words, when IoT processing and storage software is deployed to hardware, it is referred to as distribution. The Fault is decreased by employing a distributed approach since data flow and bandwidth use are minimized results in the easy communication between the devices and respective nodes.

1.2 Problem Statement

Due to the numerous variances in Internet of Things (IoT) devices topologies, morphology, environmental conditions, the identification and categorization of fault tolerant methods has proven to be a difficult challenge for specialists. Traditional identification techniques including error detection, error masking, error monitoring, and error recovery using data are time-consuming, costly and need a deep understanding of botany. The development of an accurate and effective system using Fault Tolerant Data Collection Algorithms to identify and classify various types of faults present in the IoT devices on a very large scale is necessary due to the growing demand for IoT devices and smartly connected things due to their potential health benefits in the field of medicine, safety, and the need for sustainable and responsible fault tolerant devices development practices.

This system will be created utilizing a cutting-edge fault-tolerant algorithm that can analyses the error and locate the defect using visual characteristics and patterns in the channeled data to identify and deliver simple and rapid results for the device where the error occurred. The algorithmic operations of several models for the delay and fault, together with their related labels, will be used to train the system. On the other hand, applications like permafrost monitoring demand continuous and rapid data transfer over extended periods of time, which is classified as continuous data collection.

1.3 Objectives

A key step towards delivering service quality to consumers under a variety of scenarios is the development of an effective and accurate approach for recognizing fault in nodes and in communication connections based on their interoperability. To do this, the system must be preprocessed, and fault augmentation techniques must be used to raise the accuracy of the data collection and the variety of the technical domains and data nodes. Cluster

Heads and the parent and child nodes they are related with, together with their respective labels, should be present in the Sensory Base Station nodes.
Advanced algorithms should be built to guarantee the dependable and continuous collection and data storage with failover mechanism through with rerouted data may be used. This will help design a Fault Tolerant Data Collection Network Structure model for identifying fault tolerance through optimal workload. In order to recognize multiple sorts of failures, such as node or link failure, power outages, and network congestion, among others, for fault identification jobs, these model structures may learn complicated characteristics and patterns in the fault failover process. The objective is to provide an automated, effective technique with a structural approach for correctly identifying and detecting the failover node in various communication protocols based on certain wide-ranging circumstances.

The Network Structure model must be validated using a number of validation conditions and indicators, including:
If the device is broken, verify its rank; if it turns out to be a dummy, change the topology. If it's broken, examine the number of children. Advanced fault tolerant techniques and fine-tuning the hyperparameters can boost the model's performance even more.
For example, we may focus on specific base model scenarios and their conditions where the entire structure can be adjusted to increase the precision of the fault tolerance detection task. The Cases followed are:

1. For Rank 1 is Faulty.
2. For Rank > 1 & Leaf node is Dummy.
3. For Rank > 1 & Leaf node is not Dummy.

1.4 Methodology

The parallel trees that gather data simultaneously are designed to get the data quickly from the same set of nodes to several storage station locations. It is known that the CDCT (Concurrent Data Collection Trees) network structure is not optimum in terms of the number of time slots required. In this sense, we can fairly obtain the results i.e. rings produce less data collecting time for the same number of nodes since they employ more nodes than the α -rings. Fewer time slots would be necessary for data collection if more nodes could be used in a time-slot. Here, we talk about CDCT's following drawbacks. First, the dependency that demands a maximum of one -ring if the value is required to maximize beta rings total number handled in terms of μ max is unusual. The maximum number of devices that can be allocated to a " α -ring" is also limited by CDCT. The most devices that can be connected to a " β -ring" is equal to 2 τ1 + 1. We concentrate on increasing the number of α -rings because they use more nodes than other rings do. The proposed method allows for a maximum of 2 -rings, which can be used to increase the number of rings and speed up concurrent data collecting. We seek to use more devices in each time-slot by altering the network topology by increasing the number of "rings." Additionally, the time slots 1 and 2 must be adjusted if the topology is based on maximizing the α -rings. In light of this, we suggest the updated computation of time-slots 1 and 2, which is described in the next section. The network architecture is built along with the value of max number of nodes in a data stream being even either odd, which is another CDCT constraint. We abandon this constraint and demonstrate the topology creation method regardless of whether u-max is even or odd. As a result, whether the value of maximum number of nodes is even or odd, the overall number of time-slots for concurrent data gathering in the proposed task can be decreased. The proposed method's do-able transmission schedule.

1.4 Organization

The organization's research and analysis primarily concentrate on the existence challenges between the externally induced events and actions that interfere with organizational communication functionality such as data and node communication, providing the challenges and differences with fault detection and swiftly verifying them in wireless sensing networks.

Due to obstructions or totally due to a poor connection, communication between nodes is not possible. Critical systems must operate in wireless sensor networks with reliable connectivity. Despite this, wireless sensor networks have a reputation for being unstable and prone to errors that interfere with their regular operation. Wireless sensor networks, especially in open spaces, must be capable of detecting anomalies to reduce network failures. By reducing the delays in gathering and retrieving data from wireless sensors, as well as new problems.

Network Model Development:

- Increasing the fault tolerance model's efficiency while concurrently communicating with the Base States to hasten model development and enhance performance.
- Utilize a variety of performance metrics to evaluate the model's ability to manage processes on a tree cluster and to use a hybrid model process for each running concurrent process.
- Research to determine the variables influencing model performance, depending on factors including the size of the dataset, the selection of the hyperparameters, and the number of layers in the model, various alpha or beta rings may be produced on the cluster heads.

System Validation:

- Test the system thoroughly to make sure it satisfies the required performance and functionality standards. This includes unit, integration, and acceptance testing.
- Create a procedure for continuous monitoring and feedback on errors to make sure the system is responsive to user needs and up-to-date.
- Evaluate the system's ethical and legal standing to make sure it complies with all applicable legal and ethical standards.

Documentation and Reporting:

- Create thorough documentation, such as technical specifications, user guides, and training materials, to make the project replicable and extensible by others.
- Share the project's findings and outcomes through papers, conferences, and social media to raise awareness of the endeavor and draw in IoT-related prospective collaborators.

Chapter-2

LITERATURE SURVEY

According to the research [1], the assembly of clusters of remote sensor hubs reduces energy dispersal by lowering the number of nodes involved in long-distance transmission. paper [1] focuses on assembling groups of remote sensor hubs and energy dispersing is reduced by reducing the number of nodes connected with long-distance transmission. The author proposed network design in [1] aims to reduce delays in distant sensor network information-gathering operations. Various clusters separate a group organization. in every bunch, one kind, sensor node is designated, the group leader, Cluster Head (CH), with many others serving as group members, Cluster Members (CM). The group leader will either directly or indirectly obtain information from its members. Energy dispersion is reduced by clustering remote sensor nodes and reducing the number of nodes involved in long-distance transmission. Information/choice combination at nodes along the information aggregation path can also reduce the number of information transfers and energy consumption.

The goal of a clustering algorithm is to classify sensor nodes. One node is chosen to serve as the cluster head inside each cluster. Data from cluster members must be gathered by the cluster head, combined utilizing data/decision fusion techniques and sent to the distant base station.

In terms of data collection efficiency, the proposed network structure is once again ideal, given that

i. each sensor node may only connect to one other sensor node at a time.

ii. Data fusion can be placed at any sensor node.

iii. The network has been segmented into clusters.

For the Delay Aware Proposed network structure, a tree structure is recommended. To optimize data collection efficiency, the number of nodes in the recommended network structure must be restricted to, where it will be proved in a future section that such limits may be decreased by forsaking some performance. Each member of the cluster will be allocated a rank, which is an integer between 1 and k. A node with rank k will form data links with other nodes, each of which has a different rank ranging from 1 to k-1. All of these nodes will be children of the node with rank k. A data link will be made between the node with rank and a node with a higher rank.
The figure addresses Circles with CM addressing the cluster of individuals. The circle with CH addresses the cluster head. A filled circle with BS addresses the base station. The position of every hub is addressed by the variable. The arrow addresses the presence of information connected and the heading of the arrow shows the course of the information stream.

Let’s take an example of a proposed network structure with n = 16 nodes
Individuals are addressed in circles with CM. The bunch head is addressed by a circle with CH. The base station is addressed by a filled circle with BS. The variable addresses the position of each hub.
A running bolt indicates the presence of an information link, and the bolt's heading indicates the direction of the information stream.

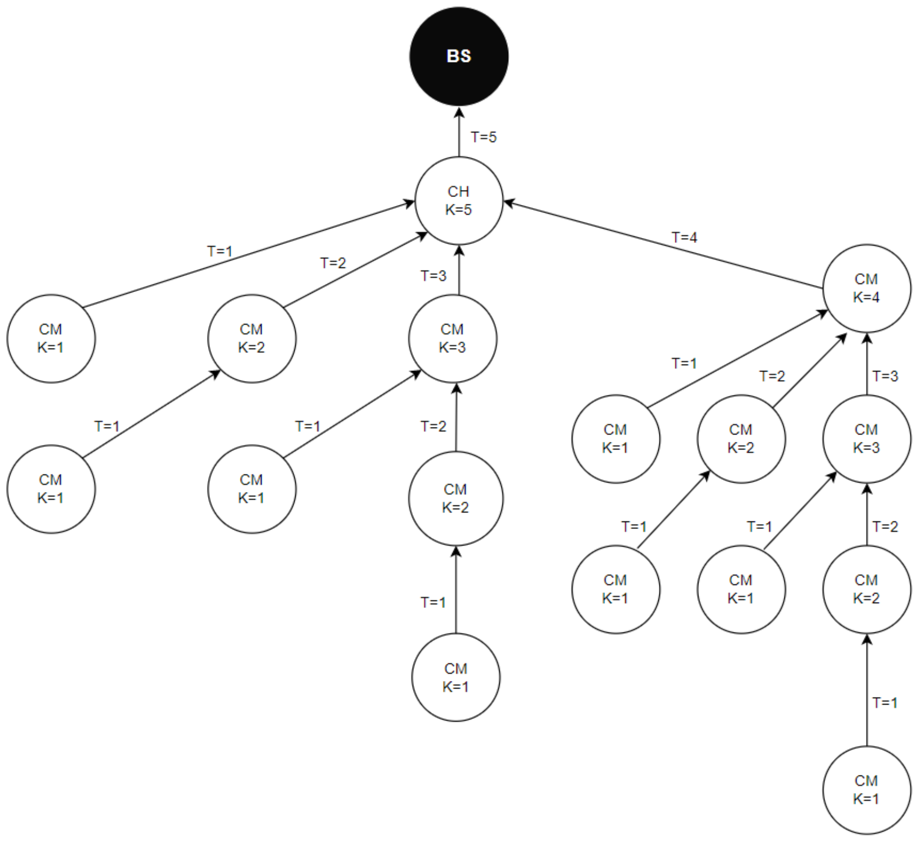


Fig. 2.1: Proposed network topology for Delay Aware Data Collection

The paper [2] aims to Trees of Concurrent Data Collection for Internet of Things Applications. It emphasizes utilizing α-ring, and β-ring structures (yield a more limited information assortment process term for a similar number of hubs). It follows Node to Node and Node to Base Station communication.

Equal information streams present new issues for IoT defer minimization. Simultaneous information assortment trees are acquainted in this article with diminishing all-out information gathering time. IoT hubs might communicate with each other and interface with BSs. The work in [2] deciphers information assembled from different IoT gadgets is accepted to be absolutely fusible, and that implies that few got information bundles might be melded into one preceding being sent to one's parent hub. A solitary unit of information will be communicated for the one-time allotment, and the term of an information combination process is accepted to be irrelevant. Each simultaneous information conglomeration activity will use an unmistakable BS to interface with the IoT organization, with a sum of k simultaneous information streams.

The procedure for creating transmission plans for the proposed network structure is basically modifiable to fit extra improvement limitations or measures. One run-of-the-mill cause of stress for portable organizations is the general correspondence distance of the information-gathering tree, which can essentially diminish the battery duration of cell phones. When the widths and number of alpha and beta rings are characterized, the proposed construction's N2N correspondence distance inside each ring can be brought down utilizing bunching procedures with given group sizes. This boundary can be additionally brought down by utilizing mobile sales rep issue solvers to change the request for the hubs inside each circle, subsequently shortening the general course length of the ring.

Power depletion in N2BS communication collisions is a disadvantage. This part of Network for Concurrent Data Collection provides two unique network topologies known as alpha ring and beta ring to achieve the desired performance in data-gathering methods. When Umax = 1, the BS of each data stream can collect information from |N| IoT nodes using star topologies (i.e., T = |N|). Data aggregation processes with durations equal to (7) may be implemented in networks with U-max = 2 and U-max = 3 by grouping the nodes into the alpha ring and beta ring, respectively.

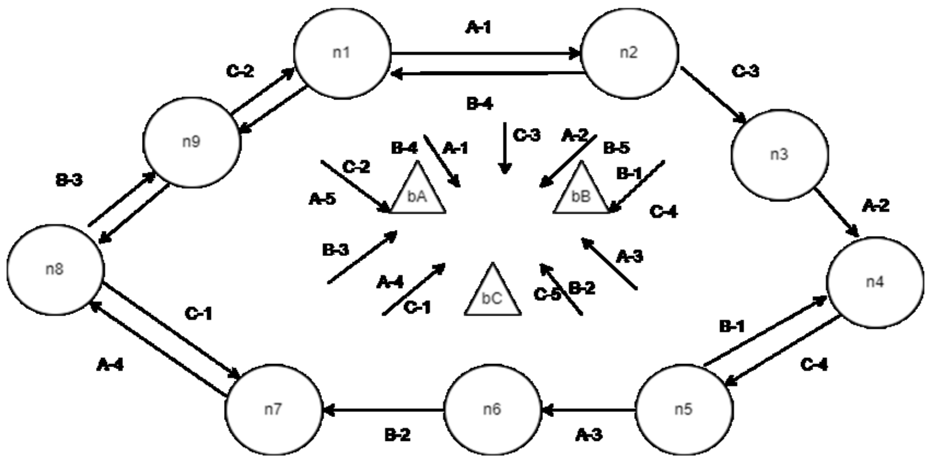


Fig. 2.2: Proposed network configuration for concurrent data collection

Data collection in a network N with |N | = 9 nodes and k = 3 concurrent data streams. Circles and triangles indicate IoT nodes and base stations, respectively. The flow of data streams is shown by arrows. The text adjacent to an arrow indicates its data stream (i.e., A, B, C) and time-slot number (i.e., 1,, 5).

The paper [3] focuses on Concurrent Data Collection Trees that are Time Optimal for IoT Applications. It utilizes more of the β-rings than α-rings. It has a combination of network topologies to lower the delays. In this research, we propose a network layout that reduces the number of time slots required for concurrent data collection.

The network configuration is made up of a group of devices/nodes, designated by N = n1; n2..., n |1|, that are shared by several applications, denoted by S = s1; s2, S|6|. We foresee a single-hop network with gadgets connecting directly to the BS. A device can also communicate with any other device in its immediate proximity. Devices can aggregate and transfer data because the data acquired by numerous devices are assumed to be linked. Multiple apps may simultaneously request data, demanding ongoing data collection. It is assumed that all N devices have data to convey and are involved in data transmission.

Because beta-rings require more nodes than alpha-rings, the article focuses on expanding the number of beta-rings. The proposed approach allows for a maximum of two alpha-rings, which enables faster concurrent data collection by increasing the beta-rings. It desired to employ more devices in each time slot by altering the network topology and increasing the beta-rings.

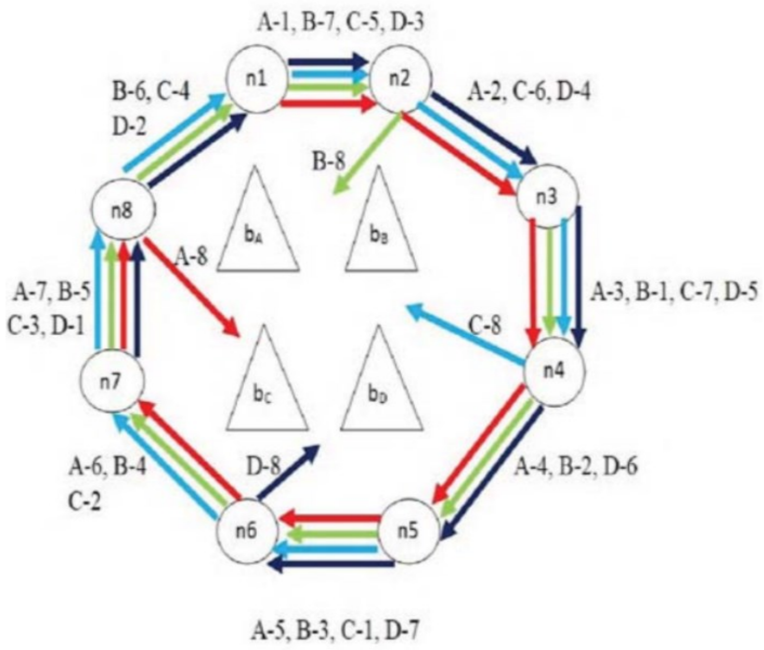


Fig. 2.3: α (Alpha)-Ring Structure

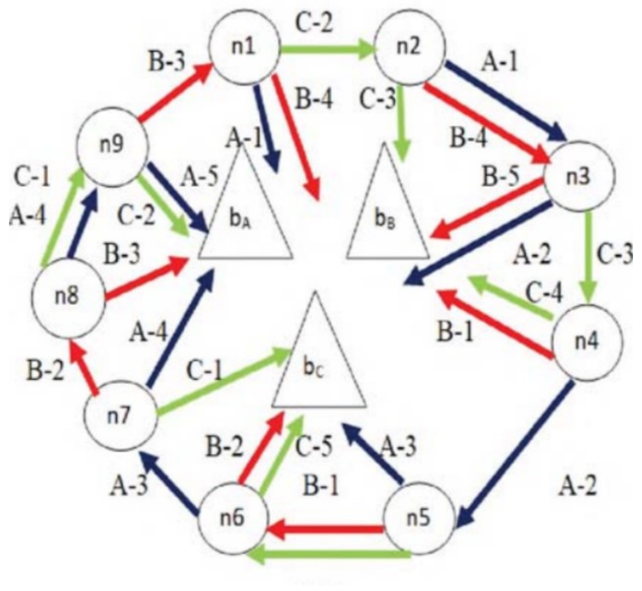


Fig. 2.4: β(Beta)-Ring Structure.

The paper [4] examines low-energy adaptive clustering hierarchy (LEACH), a convention design for microsensor networks that joins the ideas of the energy-proficient cluster-based routing and media access with application-explicit information aggregation to accomplish great framework lifetime.

LEACH incorporates another dispersed bunch development procedure that permits huge quantities of hubs to self-sort out, calculations for adjusting nodes and pivoting cluster head positions to equitably convey the energy load among all nodes, and strategies for empowering circulated signal handling to save communication assets. When contrasted with universally useful multi-hop methods, the outcomes recommend that LEACH can increase framework life expectancy by a significant degree.

In LEACH, nodes organize themselves into nearby groups, with one node filling in as the cluster head. All non-cluster head hubs communicate information to the group head, though the group head node gets information from all cluster members, performs signal handling processes on the information, and sends information to the far-off Base Station. Subsequently, being a group head node requires fundamentally more energy than being a non-cluster head hub. Assuming that the cluster heads were picked indiscriminately and stayed steady during the framework's life expectancy, these nodes would quickly drain their

restricted energy. At the point when the cluster head runs out of energy, it stops working and all cluster nodes lose communication capacity.
Subsequently, LEACH consolidates a randomized revolution of the great energy at cluster head position among the sensors to try not to deplete the battery of any one sensor in the organization. Along these lines, the energy heap of a cluster head is uniformly disseminated among the hubs.

CONCURRENT DATA COLLECTION TREES

Consider an Internet of Things (IoT) network N = n1, n2, n|N| and a collection of base stations S = s1, s2, s|S|. It is expected that each of these |N| IoT nodes can connect to the base stations and communicate with one another. Since numerous incoming data packets can be fused into one before being forwarded to one's parent node, data collected from various IoT devices is believed to be perfectly fusible [6]. A single unit of data will be transmitted throughout one time slot, and it is believed that the time it takes to combine the data will be minimal. The total number of concurrent data streams is k, and each concurrent data aggregation process will access the IoT network via a distinct base station (BS). The issue of concurrent data collecting using several data streams at multiple base stations is first addressed by the authors in Concurrent Data Collection Trees. The aforementioned method is based on the design of CDCT, which is depicted as rings, also known as α-rings and β-rings. The data collecting time from the same set of nodes to many base stations is shortened as a result of this network structure. The network structure is predicated on the idea that nodes can combine data from many IoT nodes into a single packet before sending it on. It is anticipated that a single piece of data will be conveyed once. Such data aggregation operations can all operate simultaneously to a different IoT base station. The total amount of base stations is equivalent to the total amount of active data gathering operations, which is indicated by the letter k. Additionally, it is believed that the network's transmissions are synced. In other words, numerous transmissions can take place simultaneously between non-overlapping sets of nodes. The proposed network configuration makes use of the most nodes possible during each time slot, depending on the quantity of nodes and data streams. Additionally, each data stream inside a time slot must use a unique combination of nodes for transmission. Umax specifies the maximum number of nodes that can be used by a data stream in its first time slot. All concurrent data streams should start and stop during the same time period in order to guarantee fairness among these users. However, each time slot in parallel data streams should use the same number of nodes. Each data stream should make use of the greatest number of nodes at each time-slot in order to accelerate the overall data collection process.

TIME OPTIMAL CONCURRENT DATA COLLECTION TREES

In contrast to Wireless Sensor Networks (WSNs), the Internet of Things (IoT) allows many applications to share the same device infrastructure. The devices can be queried by multiple such apps at once, which might necessitate starting concurrent data streams on the devices. The authors have noted this problem in [2]. In order to overcome this, they presented a concurrent data gathering tree structure known as α-rings and β-rings that is represented as rings. These rings are used to create the network architecture, and data is collected simultaneously at several base stations (BSs) using the same set of nodes. Therefore, it's crucial to make the most of the nodes throughout a particular time period. To do this, [4] concentrated on the β-rings rather than the α-rings to maximize node utilization. The Time Optimal CDCT network structure that minimizes the quantity of time slots needed for concurrent data collecting is defined here. The network configuration consists of a number of devices or nodes, denoted by N = {n1, n2, . . ., n|N|} ... base-stations (BS) S = {s1, s2, . . ., s|S|} are used to symbolize, n|N| that are shared by many applications. s1, s2, s3..., s|S|. We consider a single-hop network architecture in which devices can connect directly to the BS. A device can also talk to any other device in its vicinity. Assuming that the data produced by these devices is related, devices can combine and communicate data.

Concurrent data collection is necessary because multiple applications may need data at once. All devices in N are assumed to have some data to send, and are involved in data transmission. It is assumed that every device in the network N is engaged in data transmission and has some data to send. A certain number of concurrent data streams k are started in the network depending on how many of these parallel applications are requesting data. Each time-slot is typically thought of as transmitting one unit of data. The suggested network topology makes use of the most devices in the first τ1 time-slots, depending on the quantity of devices and data streams. The devices used for data transmission of various data streams vary depending on the time slot. U-max specifies the most devices that can be used by a data stream in the first time slot.

Chapter-3

SYSTEM DEVELOPMENT

3.1 Design Analysis

We have recommended, after carefully reviewing the aforementioned literature studies and other research publications, to develop a fault tolerant data gathering network topology with improved fault tolerance mechanism.

This building's design is essentially a ring of trees. When there are numerous nodes, it will be difficult to use just one design, which is why a hybrid architecture may be useful. Nodes will be arranged into clusters called "trees," each of which has a cluster head. Making the greatest use of the time that is available, the cluster heads will gather all the data from the child nodes concurrently. The cluster heads will employ ring architecture to interact with the base stations, who will then give the data to the individual processes, after collecting all the data from each group. Processes can either be greater than or less than the number of tree clusters, depending on such situation, many tree clusters will manage a certain process.

Analysis: The analysis of the methodology for fault tolerance network structure for easily communication of processes on the large scale involves evaluating the performance of the structure design and interpreting the results. If there are more processes than tree clusters, or fewer processes than tree clusters, then more than one tree cluster may be controlling a particular process. On the cluster heads, multiple alpha or beta rings may be formed, but they must stick to the final time t1 + t2. The following rules must be followed while creating the network:

There will be k concurrent processes, and a single process may be linked to several trees.

The base stations for each concurrent process will be independent.

Each tree network's cluster heads will be able to interact with one another, meaning that each cluster head may also talk to a base station.

3.2 Proposed Design

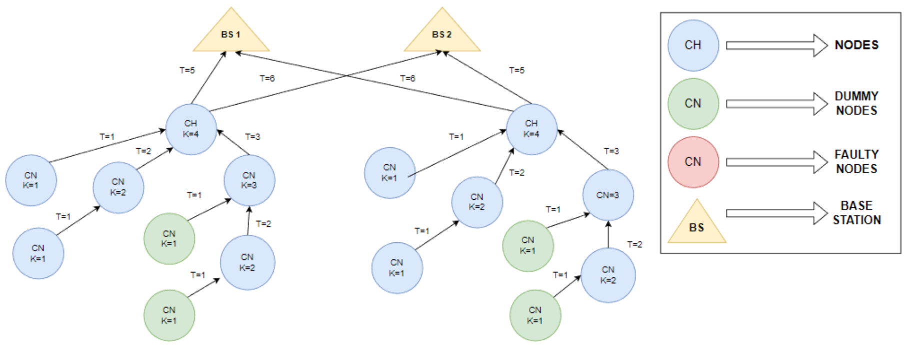


Fig 3.1: Fault Tolerant Design

Representation of Fault Tolerant Design shows the communication from:
BS-Base Stations: Serves as a Central Communication point for wireless device
CN-Cluster Node: Groups of servers that work together to perform task.
CH-Cluster Head: Gathers data from its representative child node and pass data to base stations.
Base Stations marks the communication from each cluster nodes to cluster heads to Base Station. Rank wise distribution should be provided to check the Time taken by each node from its child node to parent node.

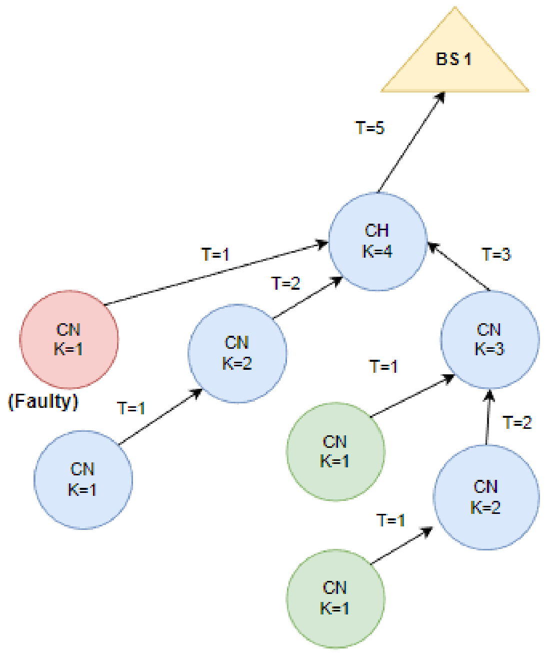


Fig 3.2: Rank=1 [Faulty]

The figure illustrates the (Cluster Node with K=1) i.e. Faulty, and Faulty node is dummy node. Dummy nodes with the Leaf Nodes [Rank=1]. Checking the Faulty node is whether dummy node.

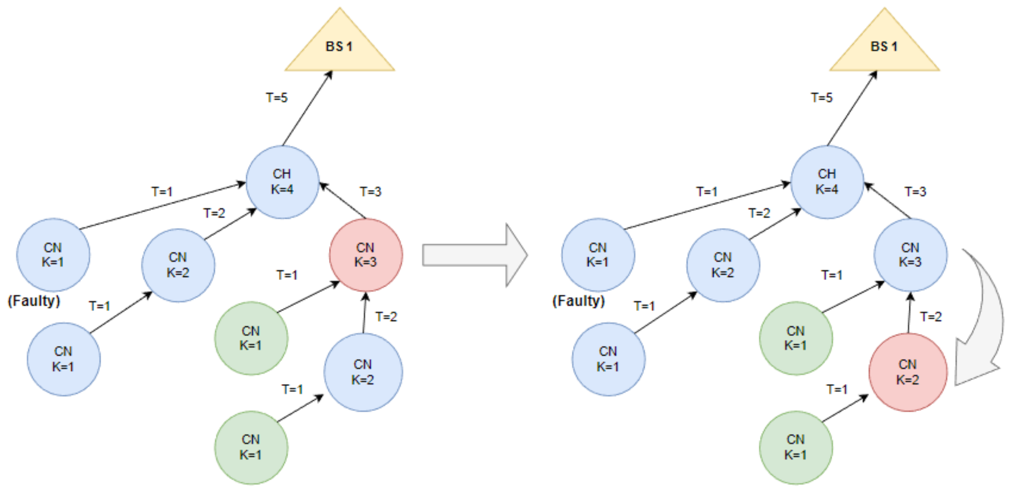


Fig: 3.3: Rank>1 & Leaf Node is Dummy

The figure illustrates the (Cluster Node with K=3) i.e. Faulty. Now, If faulty node rank >1, check if real nodes>N/2 & dummy nodes <N/2. Then replace it with a workable node. Send it to its maximum ranked child in order to remove fault.

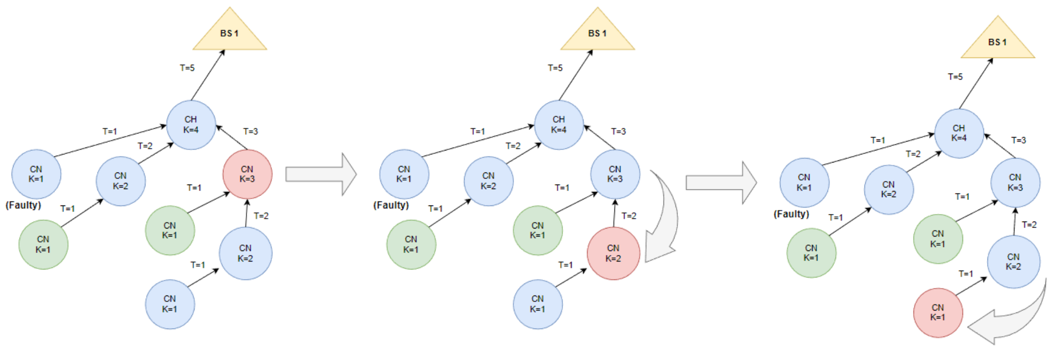


Fig 3.4: Rank>1 & Leaf Node is not Dummy

The figure illustrates the (Cluster Node with K=3) i.e. Faulty. If faulty node rank >1 & real nodes <N/2 & dummy nodes >N/2. We now restructure the whole tree. Again, starting with reconstruction of the whole structure by the faulty nodes from given nodes.

Then initializing the value of N & TN number of nodes in the form of 2^p.

We again add dummy nodes and check for the three cases mentioned above, till the whole structure if it is Fault Tolerant.

In the following figures, cluster heads link various tree clusters head to base stations and to one another. According to [3], Each tree will simultaneously collect data from its progeny for certain t1 time intervals. Following that, data will be collected simultaneously using ring architecture in t2 time periods from all cluster heads. Thus, the total time needed would be t1 plus t2.

3.3 Mathematical Analysis of Model

In a tree design, each cluster member is assigned a rank, which is a number between 1 and k. Nodes of rank k will create data linkages between nodes with rankings ranging from 1 to k-1. Each of these nodes will combine into the node with rank k, becoming its progeny. To exchange data, a node with rank k will link to a node with rank greater. The cluster leader is the network node with the highest rank. A data link will be created by the cluster leader.

The number of time slots necessary for the cluster head to collect
data from all of its offspring is provided by the equation:

$$\boldsymbol{t(N) = log_2(N) + 1}$$

In a ring architecture, there are a total of k concurrent data streams, and each concurrent data aggregation process connects to the IoT network via a different base station. All concurrent data streams should start and stop at the same time to maintain fairness among these users. A single data stream may use a maximum of |N| k nodes in a network N with k concurrent data aggregation operations during the first time slot.
In the proposed data collecting tree, a data stream would use U-max nodes in the first 1 time-slots in a row, where 1 is defined as

$$u_{max} = \left\lceil \frac{|N|}{k} \right\rceil$$

where N is the number of ring nodes and k is the number of
processes.

$$\tau\mathbf{1} = [\frac{\mathbf{2(N - u_{max})}}{\mathbf{u_{max} + 1}} + \mathbf{1}] \quad (\boldsymbol{if\ u_{max}\ is\ odd})$$

$$\tau\mathbf{1} = [\frac{\mathbf{2(N - u_{max})}}{\mathbf{u_{max}}} + \mathbf{1}], \boldsymbol{if\ u_{max}\ is\ even}$$

For $\tau\mathbf{2}$ there are the following cases:

$\tau\mathbf{2} = \{$

$$if\ (|N| - \tau 1(u_{max} + 1)/2) > 0\ and\ u_{max} is\ odd,$$
$$[log_2(|N| - \tau 1(u_{max} + 1)/2)] + 1$$

$$if\ (|N| - \tau 1(u_{max})/2) > 0\ and\ u_{max} is\ even,$$
$$[log_2(|N| - \tau 1(u_{max})/2)] + 1$$
$$otherwise,$$
$$0$$

$\}$

Therefore, the total amount of time taken would be:

$$\boldsymbol{t = \tau 1 + \tau 2}$$

Model and Assumption for Design:

Let's assume that our hybrid network has N total nodes and k concurrent processes that each have data to deliver to their respective base stations. Assume that each tree will contain T_N nodes in order to be fair, and allow there to be a total of T trees.

For each case, we must analyze, calculate the formula, and compare it to the ring architecture that already exists.

$T_N * T = |N|$ ………………. (1)

where $\mathbf{T_N}$ = node number in each tree cluster,

T = Tree Clusters number,

N = Total number of nodes in Network.

Case A: In the Case that T = n base stations:

Consider the scenario where there are as many processes as cluster heads, or the number of trees in the network, and as many base stations as there are processes.

$k = T$ (2)

$$u_{max} = T/k = 1$$

Assuming there are N total nodes, we must partition them as equally as feasible into T tree nodes, each of which has T_N nodes; T_N must strictly take the form of 2p, where p is any positive integer. From ring architecture [4] we have,

$$\tau\mathbf{1} = [\frac{\mathbf{2(T - u_{max})}}{\mathbf{u_{max} + 1}} + \mathbf{1}], \boldsymbol{if\ u_{max}\ is\ odd}$$

$$\tau\mathbf{1} = [\frac{\mathbf{2(T - 1)}}{\mathbf{2}} + \mathbf{1}]$$

$$\tau\mathbf{1} = \boldsymbol{floor(T)}$$

And the time taken to collect data by each tree is

$$t = floor(\, log_2(T_N))$$

Total time will be

$$\boldsymbol{floor(T) + floor(\, log_2(\, T_N\,))}$$

from (1) $= \boldsymbol{floor(k) + floor(\, log_2(N/k))\ \ldots}$ (3)

$\frac{N}{k} = 2^p = T_N$ ……….. (4)

Case B: when T_N will always be in the form of 2^p:

In this instance, N is taken to be selected such that N=k*2p, where p is an integer in the positive range. The graph for the following equations is shown below:

$$\boldsymbol{floor(k) + floor(log_2(N/k))}$$

$\boldsymbol{N = k * 2^p}$ from (4)

Case C: when T_N may not be presented as 2^p

In the past, we had set up N so that T_N would always be a power of two. Since N will be random in this situation, we will derive equations and evaluate them in such a way that adding fake nodes to specific tree clusters will cause T_N to be in the power of 2.

To make it acceptable for a given k and N, where N might not be in the form k*2p, we need to add a few extra fake nodes.

Total nodes that will be there after adding extra nodes will be given according to the given formula,

$\boldsymbol{N = k * 2^{ceil(log2(x/k))}}$ x=node number .. (6)

For instance, if k = 3 and x = 45, then adjusted N would be 48, which is nothing more than 3 * 24, and additional dummy nodes will only be 3.

It determines the bare minimum node that must be added.

3.4 Numerical Analysis:

Take the case where N is 100 and k is 4. The revised value of N that will be utilized to create clusters must be chosen so that the cluster tree will have 2p nodes. This is achievable by utilizing (6),

$\boldsymbol{N = k * 2^{ceil(log2(x/k))}}$ ***where x is given no of nodes.***

Formulation:

# U-maxis even:

   # τ1 = $floor$ (2 * k − 1)

   # τ2 = [log2(k * 2^p − t1 (2^(p−1)] + 1

# U-maxis odd:

   # τ1 = [2(N − $umax$)/($umax$+1) + 1]

   # τ2 = [log2(n-t1*(umax+1)/2)] +1

N = 4 * 32 = 128.

28 dummy nodes were inserted.

T (trees) = 4, and P (processes) = 4.

32 plus 2^5 is the T_N

4 base stations

Total amount of time:

$floor(k) + floor(\ log_2(N/k))$ from (3)

t = 4 + 5 = 9.

The tree will gather information from each of its children over a 5-slot period. Following that, simultaneous data transfer between the cluster heads and base stations will begin in the following manner. Let's say there are 4 processes, and A, B, C, and D the 4 cluster heads:

A will transmit to 1, B will send to 2, C will send to 3, and D will send to 4 during the first-time window.

A will transmit to 4, B will send to 3, C will send to 2, and D will send to 1. In the second time slot.

A will transmit to 2, B will send to 1, C will send to 4, and D will send to 3 in the third time slot.

A will transmit to 3, B will send to 4, C will send to 1, and D will send to 2 at the fourth time slot.

calc_umax: This returns the value of the umax parameter in integer always. It can be zero as well.

● calc_t1: This returns the t1 time as proposed in the ring architecture.

● calc_t2: This returns the t2 time as proposed in the ring architecture.

● calc_ring: Calculates the time delay using the formulas of ring architecture.

● calc_hybrid: Calculate the time delay using the formulas of the proposed design.

Algorithm:

Step 1: Initialize the value of N and k. N denotes the total number of devices in an IoT network and k is the total number of concurrent data streams.

Step 2: Initialize the value of total number of Base stations required. There will be one base station for each concurrent process.

Step 3: Let's say we have N nodes in total, we have to divide it as equally as possible into T tree nodes each having TN number of nodes, TN is strictly in the form of 2^p where p is any positive integer.

Step 4: Allocation of devices or nodes must be in form of 2^p and the time taken to collect data by each tree is t =floor(log2(TN)).

Step 5: In order to have 2^p Nodes we need to add dummy nodes.

Step 6: Now check for the inputs where fault occurred. Once fault is detected we check for three conditions (as mentioned in fig 3.3(from System Analysis)). If the faulty node is the leaf node, consider it as dummy decrement the no of real nodes and increment the no. of dummy nodes.

Step 7: If faulty node rank >1, check if real nodes>N/2 & dummy nodes <N/2. Then replace it with a workable node. Send it to its maximum ranked child in order to remove fault.

Step 8: Repeat till it reaches leaf or to a higher ranked node whose all child are dummy. Nodes should be seen out for the checking.

Step 9: If faulty node rank >1 & real nodes <N/2 & dummy nodes >N/2. We now restructure the whole tree. Again, starts with Step 1 by subtracting the faulty nodes from given nodes.
Then initializing the value of N & TN number of nodes in the form of 2^p.

We again add dummy nodes and check for the three cases mentioned above, till the whole structure if Fault Tolerant and check the Tolerances.

Model Development

Steps for Model Development Fault Tolerant Detection:

1. Import libraries: Import the necessary libraries such as NumPy, Matplotlib, NetworkX for graphical use.

2. Load data: Load the preprocessed and augmented data into the memory using various key functions.

3. Train the model: Use Node_generator method to train the model on the training node data for a specified number of values and approaches.

4. Evaluate the model: Calculate and evaluate the metrics on the various values and keys to assess the trained model's performance on the validation and testing Cluster node data. To do evaluation, use evaluate_generator method.

5. Fine-tune the model: Adjust the model's hyperparameters with variable keys such as the modern model techniques to optimize and improve the fault tolerance performance of the model.

6. Save the model: Save the trained and fine-tuned model in a format like. ipynb for later use can be use in Jupyter Notebook or Google Collaboratory.

7. Predictions: Use the trained model to make calculations on new values and process of nodes by loading the saved model and using the predict method from model by changing node values.

8. Deployment: Deploy the trained and fine-tuned model to modern IoT applications for future users of Model.

Selecting the proper hyperparameters, avoiding fault, and employing methods like basic function regularization to boost the model's performance are some modeling best practices. The performance of the model can also be validated by using real-world data from other sources.

Computational Method:

The computational method for fault tolerant detection involves several important steps.

1. Data collection: Firstly, the node data collection process is critical to the success of the system. The project team must collect the Cluster data with the nodes that are labeled with the corresponding data collection process. The dataset must be non-faulty, containing child to parent node with different backgrounds, lighting conditions to ensure that the model can generalize well to unseen data.

2. Model training: The Fault tolerant Data Collection model is trained using the training set in the model training method. During training, the

model develops the ability to identify the correct fault tolerant by separating out non-existed elements.

3. Model evaluation: Examining the model's performance on the test set comes after it has been trained. Calculating model on new metrics is necessary to determine whether the model is capable of correctly classifying fault tolerant into the appropriate model.

4. Model optimization: The model can then be further tuned via transfer learning based on the evaluation findings by the project team. In order to increase a model's accuracy and generalization performance, especially when working with fresh and new node data and Cluster heads which contains various classes of fault, this entails employing a pre-trained model and retraining it on a new data collection process.

Chapter-4

PERFORMANCE ANALYSIS

4.1 Mathematical Analysis:

When T_N can be represented as 2^p

For the hybrid model proposed the time delay is calculated as: $\boldsymbol{floor(k) + floor(\,log_2(N/k))}$ from (3)

$\boldsymbol{N = k * 2^p}$ from (5)

putting values we get, $\boldsymbol{t = floor(k) + floor(p)}$**……...** (7)

For ring architecture model proposed the time delay is calculated as:

$u_{max} = N/k = 2^p$ (even)

$\tau\mathbf{1} = [\frac{2(N - u_{max})}{u_{max}} + 1], \boldsymbol{if\ u_{max}\ is\ even}$ from [4]

$$\tau\mathbf{1} = \boldsymbol{floor(\,2 * k - 1)\ (on\ simplifying)}$$

$$\tau 2 = [log_2(|N| - \tau 1(u_{max})/2)] + 1$$

$$\tau 2 = \left[log_2(k * 2^p - \tau 1(2^{p-1})\right] + 1$$

Total time will be $\tau 2 + \tau 1$.

The proposed model and the ring architecture model have been compared and their results presented on the graph, which displays the characteristic of p in N=k*2p versus time delay. The number of processes has been maintained constant at 5 in the graph below. Our findings showed that the proposed model has a lower time delay than ring architecture and that it varies linearly with p, the power of 2, in the formula N=k*2p, where N is the number of nodes.

In N=k*2p, where N is the number of nodes, the graph illustrates the characteristic of the number of base stations or processes with the time delay at the fixed value of p, which is 3 Our findings indicate that the proposed model has a shorter time delay than the ring architecture.

4.2 Metric analysis

Case 1: where T_N can be represented in the form of 2^p

Let's start with the hybrid model approach, assuming that k = 3 and N = 3 * 24 = 48.

There will therefore be three trees, each with 24 = 16 nodes.

Considering (7), t = floor(k) + floor(p), total time is 3 + 4 = 7.

Ring architecture method: for k=3, and N = 48, $u_{max} = 2^4$.

u_{max} is even here,

$$\tau \mathbf{1} = \boldsymbol{floor}(\mathbf{2} * \boldsymbol{k} - \mathbf{1}) = \mathbf{5}.$$

$$\tau 2 = \left[log_2(k * 2^p - \tau 1(2^{p-1})\right] + 1$$

$$\tau 2 = 4$$

Total time = 5 + 4 = 9

Result: The ring architecture model has more time, while the hybrid model has less.

Case 2: when T_N cannot be represented in the form of 2^p

First using the hybrid model method:

Let's say k = 3 and N = 57.

using (7), the total nodes that will be there after adding extra nodes will be given according to the given formula,

$\boldsymbol{N = k * 2^{ceil(log2(x/k))}}$ where x is the given number of nodes (6)

$N = 3 * 2^{ceil(log2(57/3))}$

N = 96.

we need to add dummy nodes to make it 96.

$N = 3 * 2^5$

Now using formula (7), t = floor(k) + floor(p) = 3 + 5 = 8.

Second using ring model method:

k = 3, N = 57.

u_{max} = floor (N/k) = floor(19) = 19.

Since u_{max} is odd,

$$\boldsymbol{\tau 1 = [\frac{2(N - u_{max})}{u_{max} + 1} + 1], if\ u_{max}\ is\ odd}$$

$$\boldsymbol{\tau 1 = floor\ (4.8) = 4.}$$

$$\tau 2 = [log_2(|N| - \tau 1(u_{max} + 1)/2)] + 1$$

$\boldsymbol{\tau 2 = 5.}$

Total time = 5+4 = 9.

Result: As a result, the Hybrid architecture model outperforms better and relevant than Ring architecture model. Time taken by Hybrid model is also quite less as comparison with ring architecture model.

4. 3 Implementation

The proposed hybrid model and the model presented in [2], which is a ring architecture, are both implemented in the code below. We contrasted the two models based on a number of parameters and presented the results as graphs.

In the first section, we displayed the network's time delay for a fixed number of processes over a range of values for the number of nodes, N, and plotted the resulting graph. We use the value of N in the ring architecture as-is and include it into the formula suggested in the research publication [4] to obtain the time delay values. In contrast, the proposed architecture scales the value of N by adding a few dummy nodes so that $T_N = 2^p$, or the number of nodes per tree cluster, will be to the power of two.

In the second section, we plotted the graph and displayed the network's time delay for various constant values of N, or the number of nodes, over various values of K, or the number of processes. We pass N's value exactly as it is once more in the ring design, but in hybrid, we try to create dummy nodes to make it comply with the formula. We iterated over the number of processes k for some fixed value of N while keeping k at a very low level, around 10.

Chapter-5

RESULTS

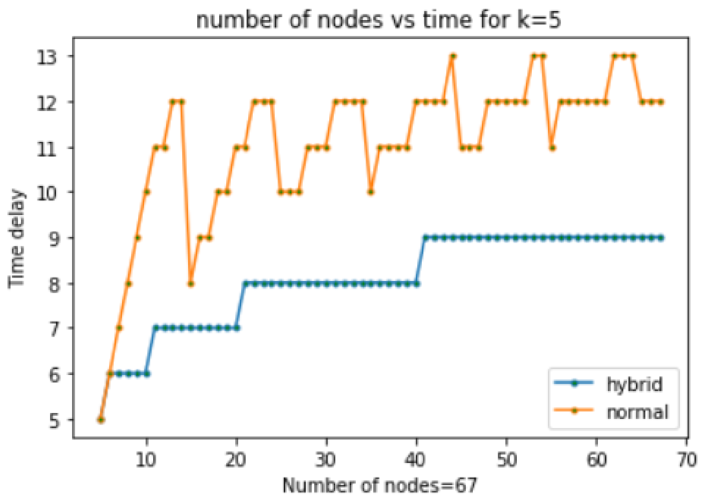
5.1 SIMULATION RESULTS:

The simulations were performed in a Python (Jupyter Notebook) The inputs of Nodes and base stations were varied and the results were then observed. The number of nodes (N) were increased in a random fashion so that the inputs do not have any correlation with each other

Comparison between number of nodes and time delay for a value-

CASE I:

```
Total number of Nodes input : 67
Input Base stations 5
---------------------------------
Total number of newNodes in the power 2^p : 80
Number of nodes in each 5 cluster : 16
Actual nodes without fault : [13, 13, 13, 14, 14]
Dummy nodes without fault : [3, 3, 3, 2, 2]
```



```
----------------------------------
Fault Detected !!!
We have to reset the structure :
----------------------------------
Total number of Nodes input : 57
Input Base stations 5
---------------------------------
Total number of newNodes in the power 2^p : 80
Number of nodes in each 5 cluster : 16
Actual nodes without fault : [11, 11, 11, 12, 12]
Dummy nodes without fault : [5, 5, 5, 4, 4]
```

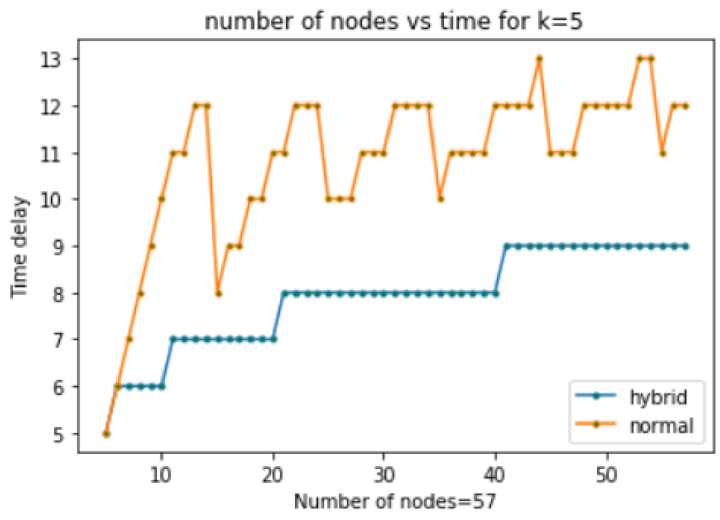
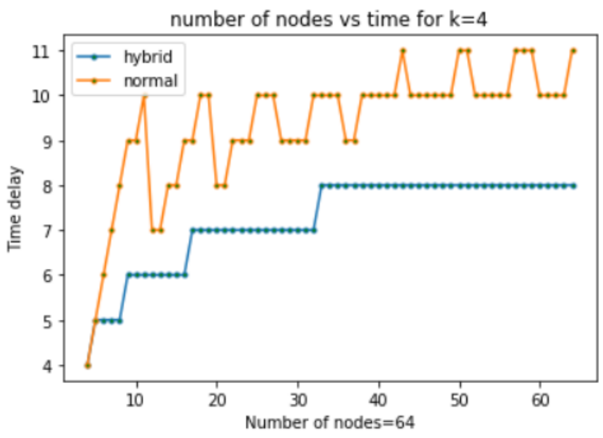


Fig. 5.1: Plot of the number of nodes vs time delay for k = 5.

Number of processes= 5 The given graph illustrates the relationship between node density and time delay. The results show that the proposed model has a lesser time delay than the ring (previous) architecture model. After detection of faulty nodes=10, Structure changes as mentioned in Fig 5.1.

CASE II:

```
Total number of Nodes input : 36
Input Base stations 4
---------------------------------
Total number of newNodes in the power 2^p : 64
Number of nodes in each 4 cluster : 16
Actual nodes without fault : [9, 9, 9, 9]
Dummy nodes without fault : [7, 7, 7, 7]
```



```
----------------------------------
Fault Detected !!!
We have to reset the structure :
----------------------------------
Total number of Nodes input : 28
Input Base stations 4
---------------------------------
Total number of newNodes in the power 2^p : 32
Number of nodes in each 4 cluster : 8
Actual nodes without fault : [7, 7, 7, 7]
Dummy nodes without fault : [1, 1, 1, 1]
```

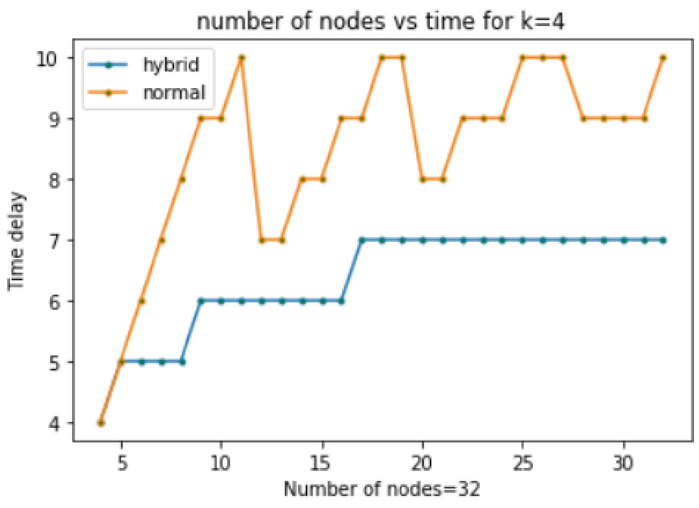


Figure: 5.2: Plot of the number of nodes vs time delay for k = 4

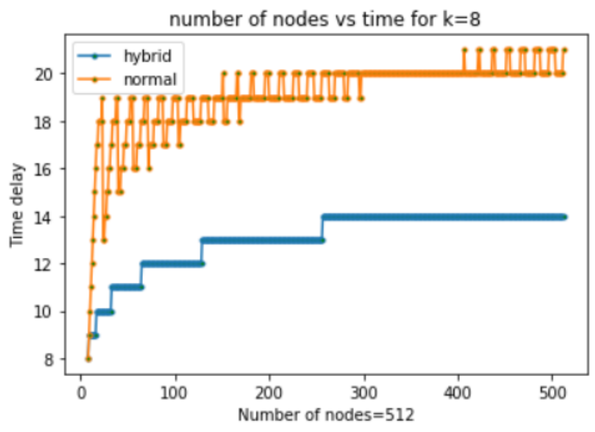
Number of processes = 4

The given graph illustrates the relationship between node density and time delay. The results show that the proposed model has a lesser time delay than the ring (old) architecture model.

After detection of faulty nodes=8, Structure changes as mentioned in Fig 5.2.

CASE III:

```
Total number of Nodes input : 300
Input Base stations 8
---------------------------------
Total number of newNodes in the power 2^p : 512
Number of nodes in each 8 cluster : 64
Actual nodes without fault : [37, 37, 37, 37, 38, 38, 38, 38]
Dummy nodes without fault : [27, 27, 27, 27, 26, 26, 26, 26]
```



```
----------------------------------
Fault Detected !!!
Only change the Topology
```

Fig. 5.3: Plot of the number of nodes vs time delay for k = 5.

Number of processes=8. The given graph illustrates the relationship between node density and time delay. The results show that the proposed model has a lesser time delay than the ring(old) architecture model. After detection of faulty nodes=2, Structure changes as mentioned in (Fig 5.3)

For the following Graphs

Comparison between number of processes and time delay for a value of N that is constant vs time delay.

CASE I:

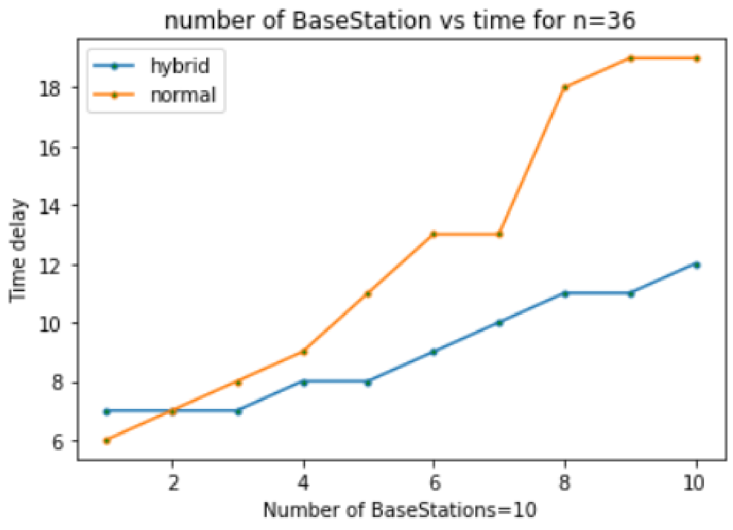


Fig. 5.4: Plot of the number of BS vs time delay N = 36.

The graph, where N is the number of nodes that were kept, displays the characteristics of the number of processes in relation to the time delay. The results show that the suggested model's time delay is less than that of the old design.

CASE II:

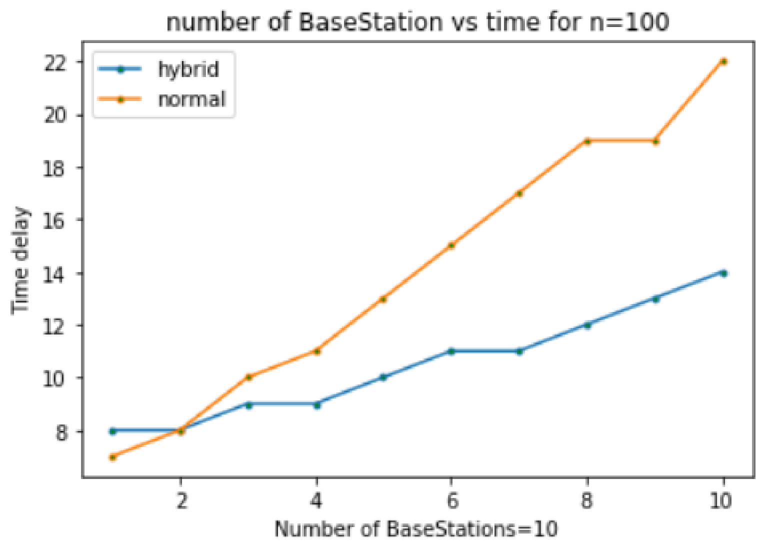


Fig. 5.5: Plot of the number of processes vs time delay N=100.

In respect to the number of nodes kept at N=100, the graph shows the features of the number of processes vs the time delay.

CASE III:

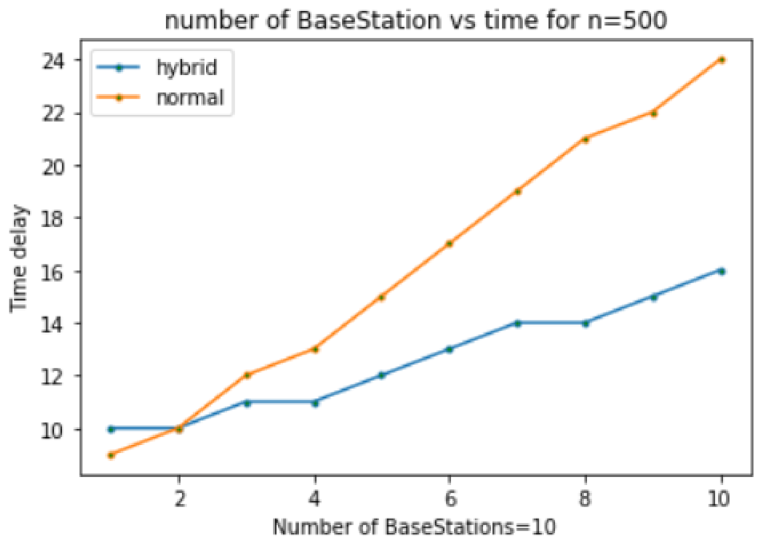


Fig. 5.6: Plot of the number of processes vs time delay for N =500.

The graph shows the features of the number of processes in relation to the time delay, where N is the number of nodes that were retained. According to the findings, the proposed model has a shorter time delay than the ring architecture model.

In this study, we suggest the best way to gather data concurrently in IoT systems. Due to the expanded advantages of shared device infrastructure, an increasing number of Internet of Things (IoT) applications rely on such shared systems for their data demands. As numerous of these applications request data simultaneously, concurrent data transmission is necessary to maintain the freshness of the data. Furthermore, quick data distribution is crucial for making crucial decisions in real-time systems. The recommended network structure in this case uses a variety of network topologies to cut down on delays. The simulation and performance analysis findings show that the suggested approach performs better than the two most often used data gathering techniques, CDCT and Time Optimal CDCT. An IoT federation is expected to be established soon as public and private internet of things (IoT) systems are linked. Under these connected systems, numerous parties will share IoT devices. Different data collecting processes launched by different users might run simultaneously on the same collection of IoT devices.

IoT devices that capture data quickly can help us obtain data with less delay than ever before. which, given that IoT applications are now commonplace in our daily lives, may eventually contribute to a worldwide gain. This endeavor may be expanded to include more fields. We have just looked at the network structure between nodes and base stations for the sake of this study. In the future, we may look at how much energy certain data collection processes use as well as how frequently each device is used in an effort to lower both. We may then use less energy to run an IoT network as a result of this. Through the processing of massive amounts of data, cloud computing is currently generating a lot of interest among various businesses where information is acquired from sensor Networks.

Chapter-6

CONCLUSIONS

6.1 COCLUSIONS

In this study, we provide the ideal method for concurrent data collection in IoT devices. A growing number of Internet of Things (IoT) applications rely on such shared systems for their data needs due to the enhanced benefits of shared device infrastructure without any fault. Concurrent data transmission is required to ensure the data's freshness since several of these apps request data at once. Furthermore, real-time systems require quick data distribution in order to make critical decisions. To reduce the fault delays, the suggested network layout in this situation employs a variety of network topologies. The results of the simulation and performance analysis demonstrate that the recommended methodology outperforms the two most popular data collection methods, CDCT and Time Optimal CDCT.

An IoT federation is expected to be established soon as public and private internet of things (IoT) systems are linked. Under these connected systems, numerous parties will share IoT devices. Different data collecting processes launched by different users might run simultaneously on the same collection of IoT devices. linked. Under these connected systems, numerous parties will share IoT devices.

Fast-capture IoT devices can assist us in obtaining data with less delay than ever before. which, given the prevalence of IoT applications in modern society, could eventually result in a global benefit. This undertaking may be broadened to include other areas. We have just looked

at the network structure between nodes and base stations for the sake of this study. In the future, we may look at how much energy certain data collection processes use as well as how frequently each device is used in an effort to lower both. We may then use less energy to run an IoT network as a result of this that capture data quickly.
Through the processing of massive amounts of data, cloud computing is currently generating a lot of interest among various businesses. where information is acquired from several sources, such as social networks, sensor networks, and automobiles. Concerns concerning the security of the data coming from the aforementioned sources and being sent to the cloud data center may still be addressed. To facilitate data gathering from sensors to the cloud, a standard architecture is required. Large heterogeneous networks of sensing devices, topologies, and protocols make up the Internet of Things (IoT). Fault identification and management is a crucial and time-consuming activity in this vast IoT network.

Network protocols act as the building blocks of any communication system that adheres to a specific Quality of Service (QoS) for each communication application. Remote system design becomes increasingly potent when installed technology advances quickly, and its topological structure and correspondence also change in unanticipated ways. It may be possible to add data collecting from WSNs and other sources to the recommended model.

6.2 FUTURE SCOPE

Future research has a huge potential for the classification of nodes inside cases that represent stationary nodes using Internet of Things (IoT) and fault-tolerant approaches for detection. The following are some possible growth areas:

1. **Increasing the Accuracy and Efficiency of the Model:** By employing larger and more thorough tolerant models, enhancing data concurrent methods, and creating more effective hardware, researchers can

continue to improve the models. The effectiveness and precision of the models can also be improved by further research and optimization of various Fault Tolerant approaches.

2. **Integrating Other Modalities:** A more thorough and precise identification of the fault tolerant model may be achieved by combining model classification function with additional modalities, such as data morphology, profiles, and genetic data methodology. Researchers can create more robust and reliable categorization systems by fusing several modalities.

3. **Developing Portable and User-Friendly Applications:** To find and detect the fault tolerant approach in the field, researchers might create applications that are simple to use for non-experts. For smartphones, tablets, and other portable devices, these applications may be created, making it simpler for consumers to obtain information about the Internet of Things and its objects.

4. **Expanding the use of Fault Tolerant Detection**: The Data Collection devices can also be used in other fields, such as the Internet of Things (IoT), Cloud Computing, and bioprospecting, in addition to the conventional ways for defect detection. Researchers can find new applications for data collecting and increase their potential advantages by investigating these new fields of use.

5. **Addressing Ethical and Legal Concerns:** The necessity to conserve traditional knowledge, stop biopiracy, and guarantee a just and equitable distribution of profits are only a few of the ethical and legal issues that should be considered as the usage of fault tolerant approaches for methodology detection increases. Involved parties and researchers can create rules and regulations to make sure moral and legal standards are upheld.

6.3 APPLICATIONS

As the world's population shifts toward relying more on technology than manual methods, everyone wants a job done for them without any effort. Everyone wants a work done for them without any effort as the world's population swings towards depending more on technology than manual techniques. The phrase "Internet of Things" essentially refers to computing equipment that transmits and receives data via the internet. The relevance of IoT in people's life is increasing as a result of its advantages and the degree of comfort people are experiencing. There are numerous ways that IoT can benefit humanity, some of which are described below:

1. ***The Medical Sector:*** Adoption on a large scale is possible. Exams, medical wearable technologies, telemedicine, and a great deal more. adoption is feasible.

2. ***Smart Homes:*** New technology has been unveiled, including Nest, Google Home, and Alexa from Amazon. Each of these devices has a certain function that improves our quality of life and makes it easier for family members to communicate online.

3. ***Intelligent Transportation Systems:*** In "smart cities," where time-wasting traffic congestion is the major issue, the Internet of Things (IoT) is providing connection and information exchange to enable proactive situational management. contemporary security and parking systems.

In addition, there are other more industries as well, including manufacturing, improved power supply, planning, industrial automation, and the digitization of cities in developing nations (for instance, have a look at Mark Zuckerberg's JARVIS). There are many opportunities.

6.4 Performance Requirements:

i. Performance Requirements:

How well a system operates under different circumstances, considering factors like time restrictions, environment, and so forth. system operates under different circumstances and considering factors.

ii. Energy Consumption:

Because the majority of Internet of Things (IoT) devices run on batteries, it is crucial to connect resource conservation to a variety of other quality factors, including performance. While coping with network connection loss, algorithms must find disconnected routes that consume the least amount of energy. most IoT devices are battery-powered, it is vital to link resource conservation to many other quality characteristics, such as performance. Algorithms must discover disjoint paths that use the least amount of energy while dealing with network connection loss.

6.5 Security Requirements:

i. Detection and Tolerance Early: The IoT network should have procedures and rules in place once an attack starts to make sure that it is stopped before it causes significant damage and extends throughout the network. IoT network architectures should be secure against hacks and other malicious assaults from the beginning.

ii. Data Transmission Enhancement & Security: Making sure data is transferred securely through a public channel without hiding information from anybody and preventing the illicit flow of information about persons or things is another aspect of security.

iii. Sensory Distribution Mechanism: This feature determines whether data analysis software should be installed on a single node or a number of nodes spread out throughout the IoT system. In other words, when IoT processing and storage software is deployed to hardware, it is referred to as distribution. The delay is decreased by employing a distributed approach since data flow and bandwidth use are minimized.

6.6 Attributes of Software Quality:

i. Accessibility: The system's capacity to function fully or partially when required. Since a fault-tolerant system is expected to operate without interruption, but a highly available system may experience service interruptions, fault tolerance and availability are not the same thing. On the other hand, a fault-tolerant approach should maintain high device availability and performance and valuable system. A fault-tolerant strategy, on the other hand, should keep device availability and performance high.

ii. Scalability: Internet of Things (IoT) systems must function effectively when there are several heterogeneous devices present. It is built on the incorporation of future on-demand resources.

iii. Localization: Processing and storage can be carried out locally or remotely depending on the amount of the data and the complexity of the needed analyses. It is built on the incorporation of future on-demand resources. The centralized cloud, dispersed edge, and fog notions start to make sense at this stage.

REFERENCES

[1] Chi-Tsun Cheng, Nuwan Ganganath & Kai-Yin Fok, "Concurrent Data Collection Trees for IoT Applications," IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 13, NO. 2, APRIL 2017.

[2] C-T. Cheng, H. Leung, and P. Maupin, "A delay-aware network structure for wireless sensor networks with in-network data fusion," IEEE Sensors J., vol. 13, no. 5, pp. 1622–1631, May 2013.

[3] C-T. Cheng and C. K. Tse, "A delay-aware network structure for wireless sensor networks with consecutive data collection processes," IEEE Sensors J., vol. 13, no. 6, pp. 2413–2422, Jun. 2013.

[4] Chi-Tsun Cheng, Chi K. Tse & Francis C. M. Lau, "A Delay-Aware Data Collection Network Structure for Wireless Sensor Networks" IEEE SENSORS JOURNAL, VOL. 11, NO. 3, MARCH 2011.

[5] J. N. Al-karaki and A. E. Kamal, "Routing techniques in wireless sensor networks," IEEE Wireless Communication Mag., vol. 11, no. 6, pp. 613–228, Dec. 2004.

[6] Zaheeruddin, D.K. Lobiyal and Aruna Pathak, "CLUSTERING SOLUTION TO MAXIMIZE LIFETIME OF WIRELESS SENSOR NETWORK", International Journal of Science, Technology & Management Volume No 04, Special Issue No. 01, March 2015.

7] T.Rashmi Anns1 , R.K.Shunmuga Priya2 , K.Mala3 , Christo Ananth. "Delay-Aware Data Collection Network Structure For WSN", International Online Conference on Advanced Research in Biology, Ecology, Science and Technology (ICARBEST'15), 19 Nov. 2015.

[8] W. Zhao and X. Tang, "Scheduling sensor data collection with dynamic traffic patterns," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 4, pp. 789– 802, Apr. 2013.

[9] N. Kapoor, S. Majumdar, and B. Nandy, “Scheduling on wireless sensor networks hosting multiple applications,” in Proc. IEEE Int. Conf. Commun., 2011, pp. 1–6.

[10] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder, “Efficient interference-aware TDMA link scheduling for static wireless networks,” in Proc. 12th Annu. Int. Conf. Mobile Comput. Netw., (MobiCom’06), Los Angeles, CA, Sep. 2006, pp. 262–273.

[11] I. Solis and K. Obraczka, “The impact of timing in data aggregation for sensor networks,” in Proc. IEEE Int. Conf. Commun., Paris, France, Jun. 2004, vol. 6, pp. 3640–3645.

[12] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless micro sensor networks,” IEEE Trans. Wireless Commun., vol. 1, no. 4, pp. 660–670, Oct. 2002.

APPENDICES

```
#/-Pseudocode for Fault Tolerant Data Collection Network-#

INPUT: number of initialNodes, BaseStations, FaultyNodes
  CALL: calculate method
  CALL: newnodenum with initialNodes,BST,faultyNodes
INITIALIZE: newNum # new nodes in power of 2
   COMPUTE: newNum = int(k * math.pow(2, math.ceil(math.log2(n/k))))
   CALL: umax with newNum,BST,faultyNodes
   COMPUTE: m # maximum utilized nodes in a cluster
   CALL: calc_umax with newNum,no of BST,faultyNodes
OUTPUT: Floor value of newNum / BST
   CALL: actualanddummy with m,BST,faultyNodes
INITIALIZE: ntemp equal to initialNodes
INITIALIZE: btemp equal to BST
INITIALIZE: act equal to list of actual nodes
    For: iteration equal to number of cluster head
INITIALIZE: 0 in act
INITIALIZE: dum equal to list of dummy nodes
    For: iteration equal to number of cluster head
INITIALIZE: 0 in dum
    FOR: iteration equal to number of cluster head
INITIALIZE: ith value of act
    COMPUTE: act[i] by ntemp/btemp typecaste to integer
INITIALIZE: ith value of dum
    COMPUTE: dum[i] by m-act[i]
INITIALIZE: ntemp
    COMPUTE: ntemp-act[i]
DECREMENT: btemp
    IF: faultyNodes>0
THEN
    CALL: condition with m,act,dum,faultyNodes
    For: iteration equal to length of dum list
    IF: dum[itr] + faultyNodes > m/2
THEN: OUTPUT: return
TRUE

ENDIF: dum[itr] + faultyNodes < m/2
THEN: OUTPUT: return FALSE
THEN: INITIALIZE: initialNodes
   COMPUTE: initialNodes equal to initialNodes-faultyNodes
           INITIALIZE: faultyNodes equal to 0
           REPEAT:
           CALL: calculate method
           UNTIL: condition OUTPUT: FALSE
```

```
ELSE:
        IF: case1 equal TRUE #RANK 1 node is faulty
  THEN: leaf node becomes faulty
ELSE: case2 equal TRUE #Rank >1 is faulty
        IF: case2_1 equal TRUE #Rank>1 leaf Node dummy
        THEN: REPEAT: replace faultyNode with maximum ranked child
        UNTIL:working higher ranked node whose all child are dummy
ELSE IF: case2_2 equal TRUE #Rank>1 leaf Node not Dummy
     THEN: replace faultyNode with leafNode

             OUTPUT: Changed the topology
      ELSE:
          OUTPUT: Working fine
```

191259_191436

ORIGINALITY REPORT

16%	10%	12%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Arvind Kumar, Rakesh Matam, Mithun Mukherjee. "Time Optimal Concurrent Data collection Trees for IoT Applications", 2021 IEEE International Systems Conference (SysCon), 2021 Publication	5%
2	core.ac.uk Internet Source	2%
3	www.coursehero.com Internet Source	2%
4	ir.juit.ac.in:8080 Internet Source	2%
5	Cheng, Chi-Tsun, Chi K. Tse, and Francis C. M. Lau. "A Delay-Aware Data Collection Network Structure for Wireless Sensor Networks", IEEE Sensors Journal, 2011. Publication	1%
6	www.iccce.co.in Internet Source	<1%