

FAULT-TOLERANCE IN IoT

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

By

Hima K S (161338)

UNDER THE SUPERVISION OF

Mr. Arvind Kumar

to

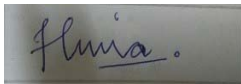


Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234,
Himachal Pradesh

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled “**Fault tolerance IoT**” in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** 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 2020 to June 2020 under the supervision of **Mr. Arvind Kumar** (Assistant Professor (Grade-II) Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



(Student Signature)

Hima K S

161338

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



(Supervisor Signature)

Arvind Kumar

Assistant Professor Grade-II

Department of Computer Science & Engineering and Information Technology

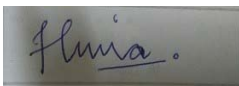
ACKNOWLEDGEMENT

First of all, I would like to express my deep gratitude to my project guide **Mr. Arvind Kumar** (*Assistant Professor (Grade II), Department of Computer Science Engineering & IT*) for providing me an opportunity to work under his supervision and guidance. His constant encouragement at every step was a precious asset during the project work.

I express my deep appreciation and sincere thanks to **Dr. Hemraj Saini** (*Department of Computer Science Engineering & IT*) for providing all kinds of possible help and encouragement during my project work.

I am thankful to the faculty and staff of *Department of Computer Science Engineering & Information Technology*, Jaypee University of Information Technology for providing me with all the facilities required for experimental work.

I would like to thank my family for their continuous support and motivation. Finally I would like to thank those who directly or indirectly helped me in completing this project.



Hima K S

TABLE OF CONTENTS

| CAPTION | PAGE NO. |
|---------------------------------------------------|-----------------|
| CANDIDATE'S DECLARATION | i |
| ACKNOWLEDGEMENT | ii |
| LIST OF ACRONYMS AND ABBREVIATIONS | v |
| LIST OF FIGURES | vi |
| ABSTRACT | vii |
| | |
| CHAPTER-1: INTRODUCTION | 1 |
| 1.1 General | 1 |
| 1.2 Problem Statement | 3 |
| 1.3 Project Objectives | 4 |
| 1.4 Methodology | 4 |
| | |
| CHAPTER-2: LITERATURE OVERVIEW | 5 |
| 2.1 A Delay Aware Data Collection Network for WSN | |
| 2.1.1 Abstract | 6 |
| 2.2 Concurrent Data Collection Trees | |
| 2.2.1 Abstract | 7 |
| 2.2.2 Concurrent Data Collection Trees | 7 |
| 2.2.3 Topologies | 9 |
| 2.2.4 Results and Analysis | 13 |

| | |
|--------------------------------------------------------------------------------------|-----------|
| 2.3 Algorithms for Fault-Tolerant Topology in Heterogeneous Wireless Sensor Networks | |
| 2.3.1 Abstract | 14 |
| CHAPTER-3: SYSTEM DEVELOPMENTS | 16 |
| 3.1 Algorithm | 16 |
| 3.1.1 Fault Tolerance Algorithm for Data Collection Trees | 16 |
| CHAPTER-4: PERFORMANCE ANALYSIS | 21 |
| 4.1 Fault Tolerance | 21 |
| 4.2 Outputs of the implemented code and their analysis | 26 |
| 4.3 Results and data set used for 3D surface plots generation | 47 |
| CHAPTER-5: CONCLUSION AND FUTURE SCOPE | 50 |
| REFERENCES | 51 |
| APPENDIX | 52 |
| PLAGIARSIM REPORT | 53 |

LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|--------|---------------------------------------------------|
| IEEE | Institute of Electrical and Electronics Engineers |
| IoT | Internet of Things |
| WSN | Wireless Sensor Network |
| CH | Cluster Head |
| CM | Cluster Member |
| DADCNS | Delay-Aware Data Collection Network Structure |
| CTP | Collection Tree Protocol |
| SC | Single Chain |
| ETX | Expected Transmissions |
| MST | Minimum Spanning Tree |
| DPP | Density Probing Packet |
| SCH | Sub Cluster Head |
| IVP | Invitation Packet |
| RP | Rejecting Packet |
| CR | Connection Request |
| MAC | Media Access Control |
| CDMA | Code Division Multiple Access |
| DCT | Data Collection Time |
| MANET | Mobile Adhoc Network |
| N2N | Node to Node |
| N2BS | Node to Base Station |

LIST OF FIGURES

DESCRIPTION

| | |
|---------|--------------------------------------|
| Fig. 1 | α Ring |
| Fig. 2 | Beta Ring |
| Fig. 3 | Fault tolerance Scenarios |
| Fig. 4 | Scenario 1 |
| Fig. 5 | Scenario 2 |
| Fig. 6 | Scenario 3 |
| Fig. 7 | Scenario 4 |
| Fig. 8 | Scenario 5 |
| Fig. 9 | Scenario 6 |
| Fig. 10 | Scenario 7 |
| Fig. 11 | Scenario 8 |
| Fig. 12 | 3D surface plot without faulty node |
| Fig. 12 | 3D surface plot with one faulty node |

ABSTRACT

Nodes in any IOT network are inclined to failure in hostile situations subsequently, in this paper, a fault tolerant algorithm is portrayed for a such network, using Concurrent Data Collection Trees. As the system becomes broken it ought to be reestablished to its typical working inside an acceptable and effective time frame. Be that as it may, if there is an occurrence of fault in Concurrent Data Collection trees, the complexity to reproduce the system increases as the quantity of parallel streams increases. Concurrent Data Collection Trees offer viable information collection and in some cases the information isn't lost totally if a flaw happens. We have talked about different cases which shed light on the information that is either totally lost or not. Two different network structures to be specific, alpha and beta rings alongside their advantages and disadvantages are talked about. An algorithm for fault tolerance has been developed for both the topologies and the algorithm has been implemented in Python.

CHAPTER 1

INTRODUCTION

1.1 General

IoT systems are picking up fame in view of their broad and efficient use in everyday exercises and their promising future applications. To help urbanization in coming years, it is critical for the citizens of the society to get cutting-edge information in an advantageous manner by grasping current advances in innovation.

Internet of Things (IoT) is an efficient arrangement among these most recent advancements. Different clients will possess and share the IoT frameworks, a course of action of sensors and middleware. Different request will be assembled by customers and even IoT devices meanwhile, which start concurrent or parallel data streams in same framework.

Sensors nodes are battery powered devices. Energy saving is very important to the lifetime of a sensor network. The total amount of energy consumed in a transmission is directly proportional to the corresponding communication distance. Hence, long distance communication between nodes and base station are usually not encouraged. The energy consumption can be reduced by adopting the clustering algorithm, where a group of clusters are formed and each cluster has its own sub-cluster head to which it send the gathered information. These sub- cluster heads in turn are connected to the cluster head which controls all clusters and finally is connected and sends data to the base station. We have carried out our work on the network structures presented by Cheng et. al. in “Delay-Aware Data Collection Network Structure for Wireless Sensor Networks” and “Concurrent Data Collection Trees for IoT Applications”.

Remote sensor systems are utilized in a variety of applications; accordingly they have various qualities which are essential for their effective working. There have been numerous kinds of systems around for remote sensors however considering Cheng's et. al. Concurrent Data Collection Trees, can improve information collection productivity as they utilize parallel information streams for information transmission.

Be that as it may, energy saving gets vital to the part of battery controlled devices. In IoT systems there are various devices working together, interconnected to one another and require a great deal of energy while working. In this manner a system must be developed so that it utilizes less power while simultaneously has less delay in transmitting information. Concurrent Data Collection Trees offer this possibility by using parallel data streams for effective data collection and slightly undermining the energy parameter. Previous network structures that communicated in parallel did not take into account both of the above variables/factors. However good they may seem, every network has its own limitations of working in hostile environments.

One such obstruction is fault in systems where we have been talked about node becomes faulty. However, in the event of parallel systems and as complex as Concurrent Data Collection Trees the task of making them fault tolerant gets difficult. In this manner, in this undertaking report we have described an effective fault-tolerant scheme for concurrent data collection trees along with various other cases that can effectively restore the networks to its normal working in a timely manner.

Other than the algorithm we have examined various cases regarding which system structure is better and the various potential outcomes of its rebuilding at various time slots related with it. Our results have been verified using simulations carried out using the code developed in Python.

1.2 Problem Statement

- To support fast development of urban communities it is significant for urban areas to convey modern data to its occupants in a timely manner.
- Among the advances, Internet of Things (IoT) is very much perceived as a promising solution.
- Currently, most existing smart urban communities are presently furnished with non-interoperable isolated IoT frameworks.
- For future IoT system, a lot of sensors and middleware will be possessed and shared by different clients.
- Users or even IoT devices may present their inquiries all the while, which trigger different parallel information streams in the same system.
- Parallel information streams acquaint new difficulties with the delay optimization in IoT frameworks.
- Therefore concurrent data collection trees are proposed to keep the general information collection term short.
- Efficient data collection processes have been very much concentrated on tangible frameworks with static topologies and single data extraction point.
- Smart devices in IoT frameworks are frequently shared by various parties; in this manner concurrent data collection processes are constantly anticipated.
- It is demonstrated that, contrasting and a existing single-client data collection structure, frameworks with the proposed tree structures can fundamentally abbreviate their concurrent data collection processes.

1.3 Project Objective

To design an algorithm for fault-tolerance for Concurrent Data Collection Tree network structure.

1.4 Methodology

We started by comparing the two data collection methods that is DADCNS (Delay Aware Data Collection Network Structure) and Concurrent data collection trees. We found that Concurrent Data Collection Trees is a more efficient method for collecting data simultaneously using parallel data streams.

However IoT devices being low power devices, we face an issue of faulty nodes. So in this project we are trying to reconstruct our network whenever any fault arises so that data is collected in a timely manner.

CHAPTER 2

LITERATURE OVERVIEW (I)

2.1 “A Delay Aware Data Collection Network Structure for Wireless Sensor Networks”

2.1.1 Abstract

In any system to save energy is extremely critical. Here in this paper i.e. "Delay aware data collection network structure for wireless sensor networks" is proposed, two algorithms are suggested and actualized to manage this very issue of limiting energy to increase longer life time, for that even the efficiency of gathering information is ignored.

Wireless sensor networks use enormous quantities of wireless sensor nodes to gather data from their detecting territory. Wireless sensor nodes are battery-fueled devices. Energy saving is constantly critical to the lifetime of a wireless sensor network. As of late, numerous algorithms are proposed to handle the energy saving issue in wireless sensor systems. In these algorithms, in any case, information collection efficiency is typically undermined in return for increasing longer system lifetime. There are solid needs to create wireless sensor network algorithms which take care of optimization priorities other than energy saving.

In this paper, a delay-aware data collection network structure for wireless sensor networks is proposed. The target of the proposed system structure is to limit delays in the data collection procedures of wireless sensor networks. Two network development algorithms are designed to build the proposed system structure in a brought together and a decentralized methodology. Performances of the proposed system structure are assessed utilizing computer simulations. Simulations results show that, when contrasting with other common network structures in wireless sensor network, the proposed system structure can abbreviate the delay in the data collection process altogether.

LITERATURE OVERVIEW (II)

2.2 “Concurrent Data Collection Trees for IoT Applications”

2.2.1 Abstract

In IoT networks many devices work together. They create huge measures of data subsequently data collection process turns into a central worry in large systems. Data collection processes must use least measure of time while gathering information. Hence another system structure – Concurrent Data Collection Trees for IoT systems has been proposed in this paper. This system limits information assortment time to a acceptable value. Another worry is that if a large system, as huge as IoT network experiences any issue. Base Station or a node in a system may become flawed on account of numerous reasons – natural conditions, battery run out, inappropriate taking care of, and so on. To take care of this issue we are attempting to plan a methodology for adaptation to internal failure in – Concurrent Data Collection Trees network structure.

2.2.2 Concurrent Data Collection Trees

It is a network structure $N = \{n_1, n_2, n_3, \dots, n_N\}$ nodes and $S = \{s_1, s_2, s_3, \dots, s_S\}$ base stations assuming that all IoT nodes communicate with each other and base station. Data fusion strategy fuses numerous packets of information (from IoT nodes) into one single packet before it is sent to one's parent node. For concurrent data collection processes it must utilize equivalent number of data streams and base stations – "Each concurrent data aggregation process will utilize an alternate base station (BS) to get to the IoT network and total number of parallel data streams is k". Concurrent data streams must utilize equivalent number of nodes dictated by condition (1). This condition decides most extreme number of streams a node must use with the goal that data collection time is minimum.

This networks structure has a constraint: $|N| \geq k$

The following equation represents number of hubs used by an information stream in i^{th} vacancy.

$$u_{max} = \text{floor}(|N| / k) \quad (1)$$

$$u_i = \min[u_{max}, |N| - \sum_{j=1}^{i-1} \hat{u}_j] \quad (2)$$

where \hat{u}_j is number of nodes that have completed transmission of data after j^{th} time slot.

If u_i is odd then one of the nodes is involved in node-to-base transmission, else it is a node-to- node transmission.

Time-slot is a particular slot in which each data stream helps to communicate with nodes and base stations. During a particular time slot only some nodes (maximum 3) and some concurrent data streams become active. This is what justifies the parallel behavior of the given network structure.

In T1 time slot all nodes and base stations communicate in concurrent fashion, but in T2 time slot the left over nodes communicate using DADCNS.

The leftover nodes are calculated using $|N| - \tau_1[u_{max}]$.

$$\tau_1 = \begin{cases} \lfloor \frac{2(|N| - u_{max})}{(u_{max} + 1)} + 1 \rfloor, & \text{if } u_{max} \text{ is odd,} \\ \lfloor \frac{2(|N| - u_{max})}{u_{max}} + 1 \rfloor, & \text{if } u_{max} \text{ is even.} \end{cases}$$

$$\tau_2 = \begin{cases} \lfloor \log_2(|N| - \tau_1 \frac{u_{\max}+1}{2}) \rfloor + 1, & \text{if } |N| - \tau_1 \frac{u_{\max}+1}{2} > 0 \\ & \text{and } u_{\max} \text{ is odd,} \\ \lfloor \log_2(|N| - \tau_1 \frac{u_{\max}}{2}) \rfloor + 1, & \text{if } |N| - \tau_1 \frac{u_{\max}}{2} > 0 \\ & \text{and } u_{\max} \text{ is even,} \\ 0, & \text{otherwise.} \end{cases}$$

Therefore overall duration of k concurrent data collection processes is $T = T_1 + T_2$.

2.2.3 Topologies

A) The α -ring

A ring structure for concurrent data collection with $|N_\alpha|$ nodes and $|N_\alpha| \geq 2k$. An α -ring case is legitimate just for $u_{\max} = 2$. "A data stream in a α -ring N_α will require T_1 time slots to aggregate information from $|N_\alpha|$ nodes onto a single node. Such a node will take one time slot to report the fused information to the BS". On the off chance that nodes are numbered arbitrarily, at any point whenever two nodes that will communicate (during K th data collection process) i.e. node nc_1 transmits information to node nc_2

$$c_1 = (1 + \text{mod}(2(K-1) + t-1, |N_\alpha|)),$$

$$c_2 = (1 + \text{mod}(2(K-1) + t, |N_\alpha|))$$

Data fusion will be performed on node n_2 . Overall duration can be calculated using T_1 and T_2 from equations (2) &(3)

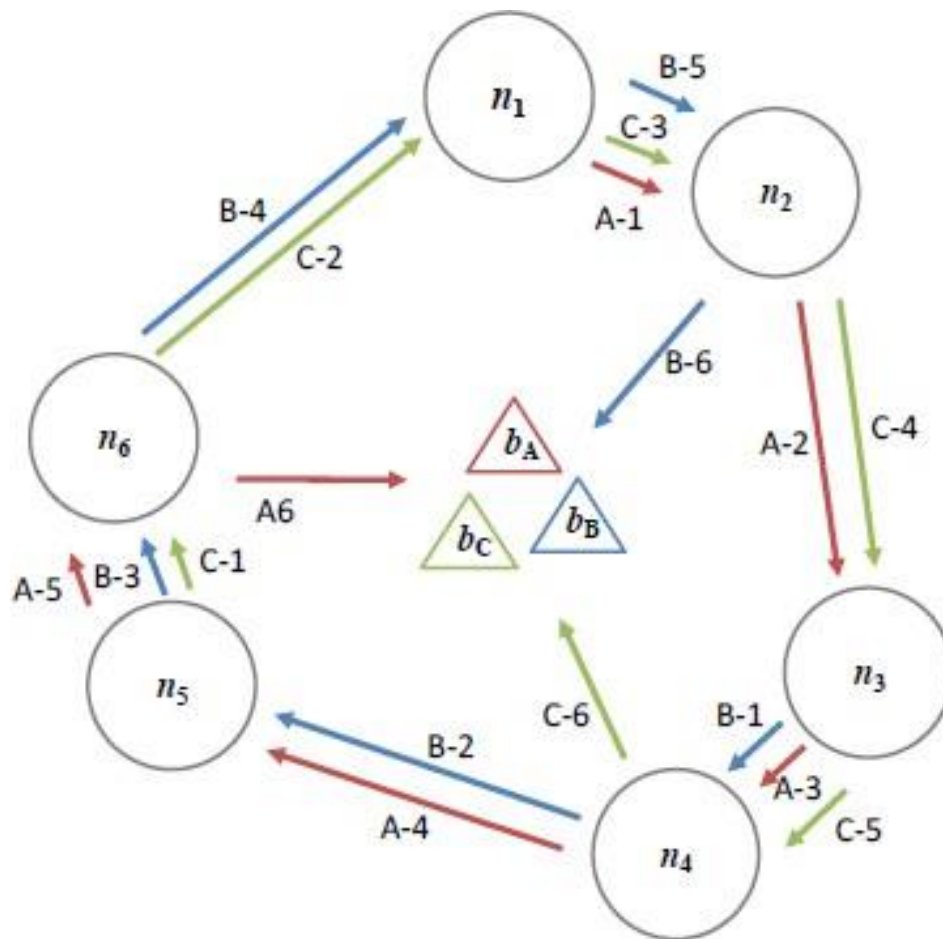


Fig 1 Alpha ring

B) The β -ring

For $u_{\max} = 3$ and $|N\beta| \geq 3k$ the network is known as β -ring. 2 nodes will communicate utilizing node to-node (N2N) communication while the other node will communicate utilizing node to-base (N2BS) communication. In the event that nodes are numbered subjectively, at that point at a specific time slot of K th data collection process node $nc3$ will be engaged with node to-base station communication and $nc4$ will transmit data $nc5$.

$$c3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N\beta|))$$

$$c4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N\beta|))$$

$$c5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N\beta|))$$

Data fusion will be performed on node $nc5$.

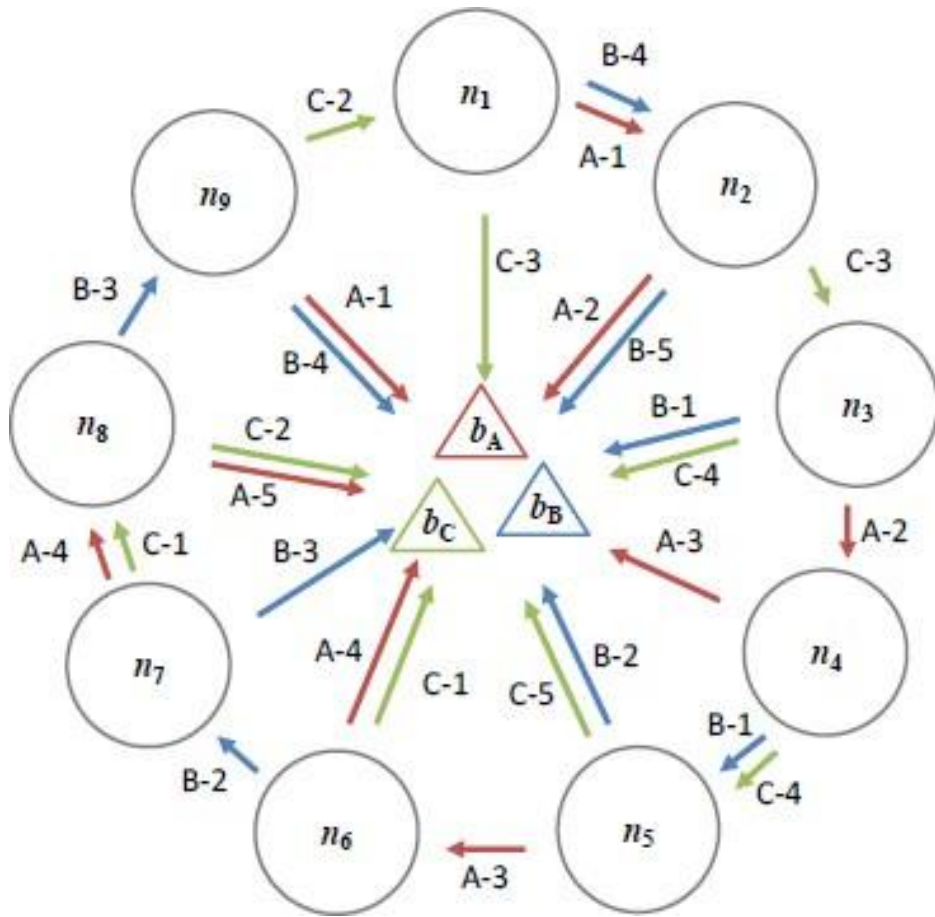


Fig. 2 Beta ring

$$T = T1 + T2 = 4 + 1 = 5.$$

C) Multiple Rings

I) $u_{max} \geq 4$: $n\alpha = u_{max}/2$ α -rings are formed. Each of the α -rings formed will initially be allowed $2k$ nodes. The rest $|N| - n\alpha(2k)$ nodes will at that point be granted to those $n\alpha$ rings individually. Each α -ring will work autonomously as depicted previously.

II) $u_{max} \geq 5$: A single β -ring with $n'\alpha = (u_{max}-3)/2$ number of α -rings are formed. At first, β -ring will be allowed $3k$ nodes, while each α -ring will be conceded $2k$ nodes. The rest $|N| - 3kBS - n'\alpha(2kBS)$ node will be allowed to the β -ring until $|N\beta| = 2f1 + 1$. The rest of the nodes will be circulated to α -rings individually.

2.2.4 Result and Analysis

The duration of data collection process T (absolute number of spaces required by base stations of various data streams) is utilized as a performance indicator. The reference structure utilized for performance comparison is DADCNS. DADCNS is sued in form of a single cluster. The performance parameter of concurrent data collection tree network structure is relatively low in contrast with DADCNS.

- As k increases, the value of T increments linearly in DADCNS while in the concurrent data collection tree structure with increment in $|N|$, estimation of u_{max} likewise increases but the total value of T increases slightly. In this way with increment in $|N|$ and k the performance gap between two system structures likewise enlarges.
- When value of k or $|N|$ is expanded, estimation of u_{max} additionally changes. This adjustment in the estimation of u_{max} prompts variations in the number of α and β rings. Accordingly it is seen that when k or $|N|$ increases value of T doesn't increment monotonically.

LITERATURE OVERVIEW (III)

2.3 “Fault tolerance in Concurrent Data Collection Trees”

2.3.1 Abstract

In this paper, we address topology control in heterogeneous wireless sensor networks (WSNs)

comprising of two kinds of remote devices: asset compelled wireless sensor nodes conveyed arbitrarily in a huge number and an a lot more smaller number of resource rich supernodes put at known locations. The supernodes have two transceivers: one interfaces with the WSN, and the other associates with the supernode network. The supernode network gives better QoS and is utilized to rapidly advance sensor information packets to the client. With this setting, information assembling in heterogeneous WSNs has two stages. To begin with, sensor nodes transmit and relay measurements on multi-hop ways toward any supernode. At that point, when an information packet is forwarded to a supernode, it is sent utilizing quick supernode-to-supernode communication toward the client application. Moreover, supernodes could process sensor information before sending.

An examination by Intel shows that utilizing a heterogeneous architecture brings about improved network performance, for example, a lower information gathering delay and a more extended system lifetime. Hardware parts of the heterogeneous WSNs are currently economically available. We model topology control as a range task issue, for which the communication scope of every sensor node must be processed. The goal is to limit the absolute transmission control for all sensors while keeping up k -vertex disjoint communication ways from every sensor to the set of supernodes. Along these lines, the system can endure the failure of up to $k - 1$ sensor nodes. Conversely with extend task in specially appointed remote systems; this issue isn't worried about connectivity between any two nodes.

Our concern is explicitly custom fitted to heterogeneous WSNs, in which information is sent from sensors to supernodes. The commitments of this paper are the following: 1) we formulate the k -degree Anycast Topology Control k -ATCP issue for heterogeneous WSNs, 2) we propose three answers for taking care of the k -ATC issue, an) a k -approximation algorithm, b) a concentrated greedy algorithm that limits the sensor most extreme transmission range, and c) a distributed and localized, and 3) we examine the performance of these algorithms through simulations.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Fault Tolerance Algorithm

A) Fault tolerance algorithm for concurrent data collection trees:

When single node becomes faulty :

Start

Calculate $u_{max} = \lfloor n/k \rfloor$

$u_{temp} = u_{max}$

Calculate T1,T2

Calculate nAlpha, nBeta

Calculate nAlphaNodes, nBetaNodes

$nAlpha_{temp} = nAlpha, \quad nBeta_{temp} = nBeta$

$nAlphaNodes_{temp} = nAlphaNodes, \quad nBetaNodes_{temp} = nBetaNodes$

while (1)

{

if($N_{new} == N$)

{

Calculate $new_u_{max} = \lfloor n-1/k \rfloor$

Calculate T1,T2

If $new_u_{max} == u_{temp}$

 If $u_{max} \% 2 == 0$

 If fnode exists in alpha max

Delete(fnode)

Else if fnode exists in alpha min

Delete(fnode)

new=alphamax.lnode

insert new in alphamin end

delete (alphamax.lnode)

Else

if $N > 3k$

If (nBetaNodes < nBetaNodes_temp)

Delete(fnode)

If(nBetaNodes < 3k)

Check alpha rings with $> 2k$ nodes

new=alphamax.lnode

insert new in beta end

delete (alphamax.lnode)

2. Else

If(alphaMax==alphaMin)

Delete(fnode)

```
if(nAlphaNodes<2k*nAlpha)
```

```
  a. new=beta.lnode
```

```
      b. insert new in alphamin end  
         (beta.lnode)
```

```
      c. delete
```

```
  Else if fnode exists in alpha max
```

```
    Delete(fnode)
```

```
  Else if fnode exists in alpha min
```

```
    Delete(fnode)
```

```
    new=alphamax.lnode
```

```
    insert new in alphamin end
```

```
    delete (alphamax.lnode)
```

```
Else // umax changes
```

```
  If utemp %2 !=0
```

```
    delete(fnode)
```

```
    reconstructA(N-1)
```

```
  else
```

```
    delete(fnode)
```

```
    reconstructB(N-1)
```

```
}
```

```
}
```

```
Delete(fnode)
```

```
If fnode->next==NULL
```

```
    Fnode=null
```

```
Else
```

```
    fnode->next->prev=fnode->prev
```

```
    fnode->prev->next=fnode->next
```

```
    fnode=null
```

```
End
```

- fnode: faulty node
- enode:extra node
- cnode:child node
- lnode:last node

CHAPTER 4

Performance Analysis

4.1 Fault Tolerance (Theoretical explanation)

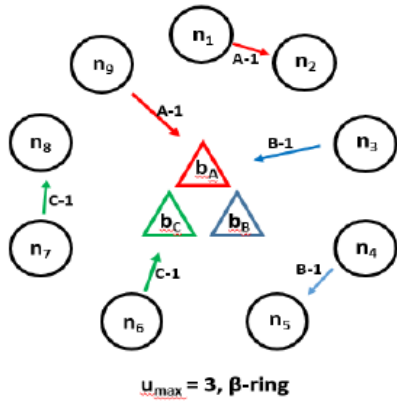
a) **Alpha Ring:**

Assume as given in the figure beneath, the fault happens during some underlying time slot then there is either practically no loss at all and the system can be reproduced once more. This outcomes in less time complexity since the system where the issue happens is recreated promptly accordingly reestablishing its typical working. In any case, in the intermediate time slots, a fault may bring about noteworthy loss of information with it remunerating to recreate it from the beginning.

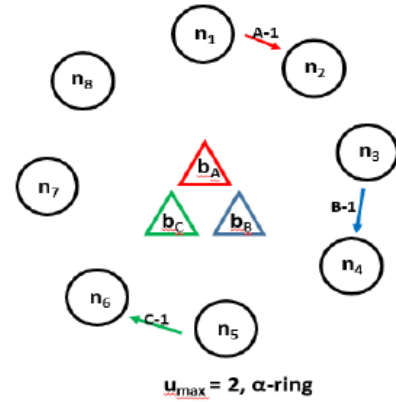
b) **Beta Ring:**

In this system the information loss is altogether more in contrast with alpha ring as certain nodes have already transferred information to the base station and during failure even the information aggregation nodes are not ready to completely recoup the information as they are loosely associated on account of their network communication structure.

NODE FAULTY

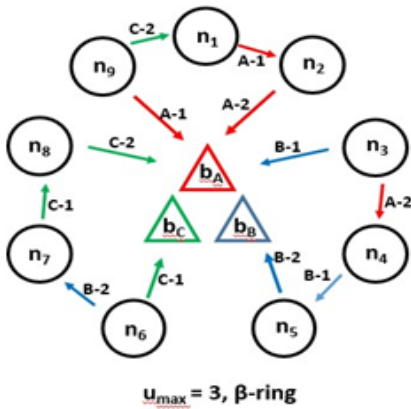


→ n_9 becomes faulty →

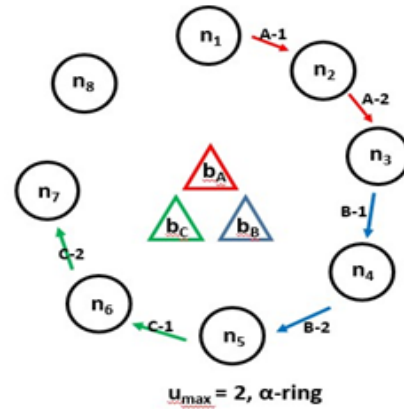


Beta Ring with 9 nodes Ist time slot
 $K=1, 2, 3. \quad t=1. \quad N_B=9$
 $c3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N_B|)) = 1, 4, 7$
 $c4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N_B|)) = 2, 5, 8$
 $c5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N_B|)) = 3, 6, 9$

Alpha Ring with 9 nodes Ist time slot
 $K=1, 2, 3. \quad t=1. \quad N_A=8$
 $c1 = (1 + \text{mod}(2(K-1) + t-1, |N_A|)) = 1, 3, 5$
 $c2 = (1 + \text{mod}(2(K-1) + t, |N_A|)) = 2, 4, 6$

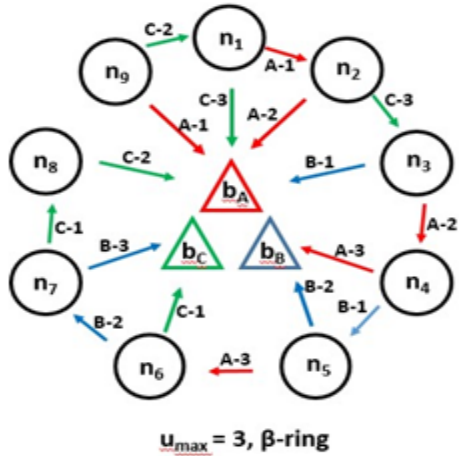


→ n_9 becomes faulty →



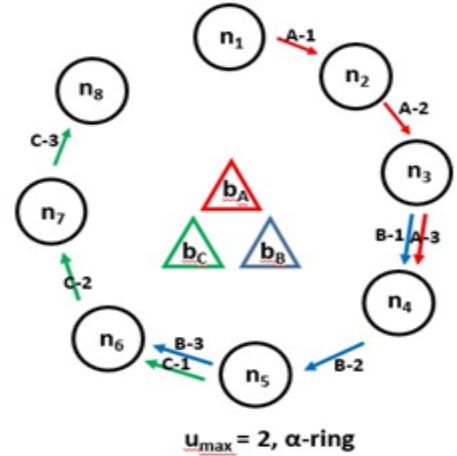
Beta Ring with 9 nodes IInd time slot
 $K=1, 2, 3. \quad t=2. \quad N_B=9$
 $c3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N_B|)) = 3, 6, 9$
 $c4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N_B|)) = 4, 7, 1$
 $c5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N_B|)) = 2, 5, 8$

Alpha Ring with 9 nodes IInd time slot
 $K=1, 2, 3. \quad t=2. \quad N_A=8$
 $c1 = (1 + \text{mod}(2(K-1) + t-1, |N_A|)) = 2, 4, 6$
 $c2 = (1 + \text{mod}(2(K-1) + t, |N_A|)) = 3, 5, 7$

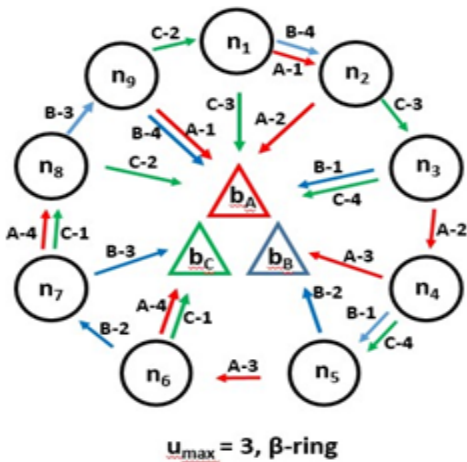


Beta Ring with 9 nodes IIIrd time slot
 $K=1, 2, 3. \quad t=3. \quad N_\beta=9$
 $c_3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N_\beta|)) = 5, 8, 2$
 $c_4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N_\beta|)) = 6, 9, 3$
 $c_5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N_\beta|)) = 4, 7, 1$

n_9 becomes faulty

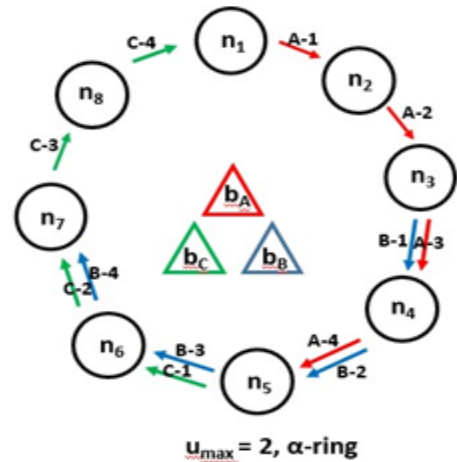


Alpha Ring with 8 nodes IIIrd time slot
 $K=1, 2, 3. \quad t=3. \quad N_\alpha=8$
 $c_1 = (1 + \text{mod}(2(K-1) + t - 1, |N_\alpha|)) = 3, 5, 7$
 $c_2 = (1 + \text{mod}(2(K-1) + t, |N_\alpha|)) = 4, 6, 8$

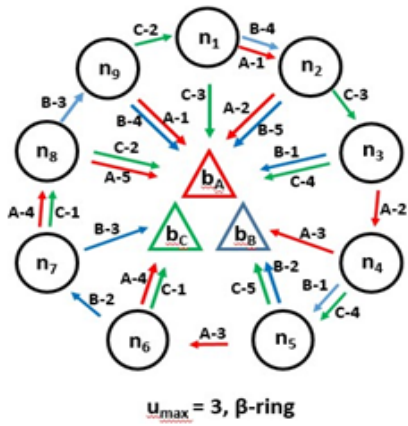


Beta Ring with 9 nodes IVth time slot
 $K=1, 2, 3. \quad t=4. \quad N_\beta=9$
 $c_3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N_\beta|)) = 7, 1, 4$
 $c_4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N_\beta|)) = 8, 2, 5$
 $c_5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N_\beta|)) = 6, 9, 3$

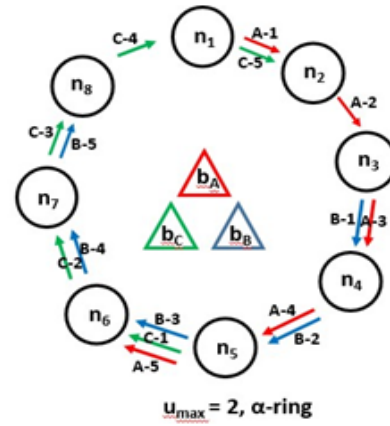
n_9 becomes faulty



Alpha Ring with 8 nodes IVth time slot
 $K=1, 2, 3. \quad t=4. \quad N_\alpha=8$
 $c_1 = (1 + \text{mod}(2(K-1) + t - 1, |N_\alpha|)) = 4, 6, 8$
 $c_2 = (1 + \text{mod}(2(K-1) + t, |N_\alpha|)) = 5, 7, 1$

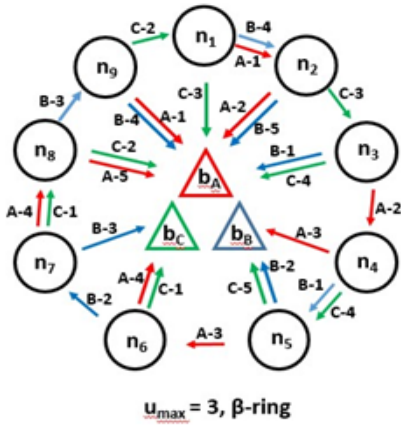


$\xrightarrow{n_9 \text{ becomes faulty}}$

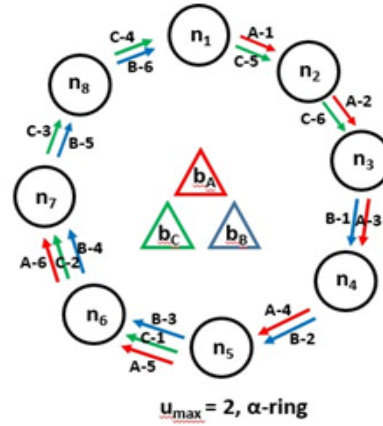


Beta Ring with 9 nodes V^{\downarrow} time slot
 $K=1, 2, 3. \quad t=5. N_p=9$
 $c_3 = (1 + \text{mod}(3(K-1) + 2(t-1), [N_p])) = \phi$
 $c_4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, [N_p])) = \phi$
 $c_5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, [N_p])) = 8, 2, 5$

Alpha Ring with 8 nodes V^{\downarrow} time slot
 $K=1, 2, 3. \quad t=5. N_a=8$
 $c_1 = (1 + \text{mod}(2(K-1) + t-1, [N_a])) = 5, 7, 1$
 $c_2 = (1 + \text{mod}(2(K-1) + t, [N_a])) = 6, 8, 2$

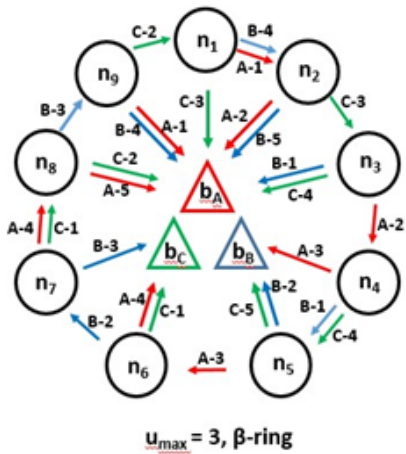


$\xrightarrow{n_9 \text{ becomes faulty}}$

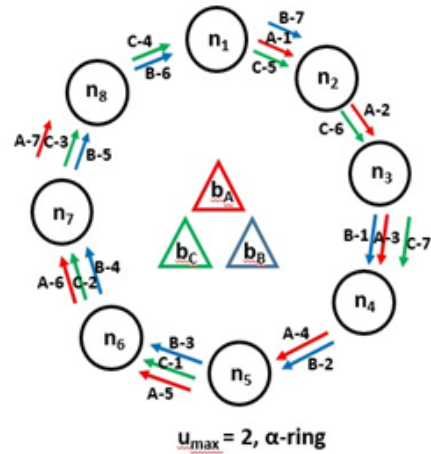


Beta Ring with 9 nodes V^{\downarrow} time slot
 $K=1, 2, 3. \quad t=5. N_p=9$
 $c_3 = (1 + \text{mod}(3(K-1) + 2(t-1), [N_p])) = \phi$
 $c_4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, [N_p])) = \phi$
 $c_5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, [N_p])) = \phi$

Alpha Ring with 8 nodes VI^{\downarrow} time slot
 $K=1, 2, 3. \quad t=6. N_a=8$
 $c_1 = (1 + \text{mod}(2(K-1) + t-1, [N_a])) = 6, 8, 2$
 $c_2 = (1 + \text{mod}(2(K-1) + t, [N_a])) = 7, 1, 3$

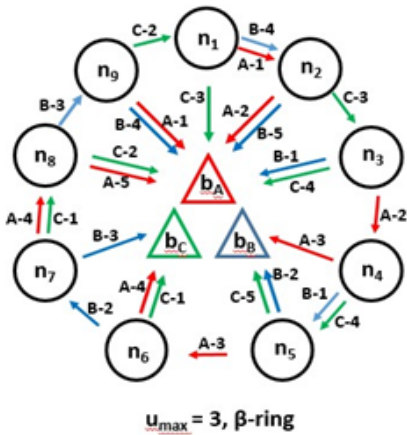


$\xrightarrow{n_9 \text{ becomes faulty}}$

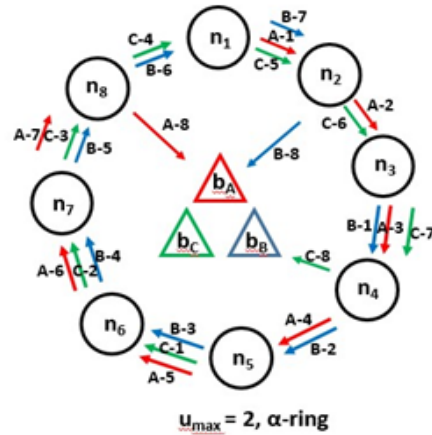


Beta Ring with 9 nodes V^{th} time slot
 $K=1, 2, 3$. $t=5$. $N_{\beta}=9$
 $c_3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N_{\beta}|)) = \phi$
 $c_4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N_{\beta}|)) = \phi$
 $c_5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N_{\beta}|)) = \phi$

Alpha Ring with 6 nodes VII^{th} time slot
 $K=1, 2, 3$. $t=7$. $N_{\alpha}=8$
 $c_1 = (1 + \text{mod}(2(K-1) + t - 1, |N_{\alpha}|)) = 7, 1, 3$
 $c_2 = (1 + \text{mod}(2(K-1) + t, |N_{\alpha}|)) = 8, 2, 4$



$\xrightarrow{n_9 \text{ becomes faulty}}$



Beta Ring with 9 nodes V^{th} time slot
 $K=1, 2, 3$. $t=5$. $N_{\beta}=9$
 $c_3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N_{\beta}|)) = \phi$
 $c_4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N_{\beta}|)) = \phi$
 $c_5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N_{\beta}|)) = \phi$

Alpha Ring with 6 nodes $VIII^{\text{th}}$ time slot
 $K=1, 2, 3$. $t=8$. $N_{\alpha}=8$
 $c_1 = (1 + \text{mod}(2(K-1) + t - 1, |N_{\alpha}|)) = 2, 4, 8$
 $c_2 = (1 + \text{mod}(2(K-1) + t, |N_{\alpha}|)) = \phi$

4.2 Analysis of implementation of the algorithm with examples and outputs:

- ❖ Cases where u_{\max} does not change: No change in u_{\max} means no change in the number of α -rings or β -rings.

Example-1:

- $|N|=25$ $k=4$ $u_{\max} = \lfloor 25/4 \rfloor = 6$

- Since u_{\max} is even, only α -rings will be formed.
- Number of α -rings: $(u_{\max} / 2) = 3$
- Each α -ring will be assigned with $2k$ nodes first:

8 8 8 (Total nodes:24)

- This value only adds up to 24, the remaining node will be assigned to the first α -ring:

9 8 8 (Total nodes:25)

- This structure contains one α -max, which is the first α -ring among the three rings.
- Rest of the two α -rings are α -min.

- Using the formula, and calculating timeslots, T1 and T2:

T1:7 T2:3 T1+T2:10

Single node fault occurs:

Cases:

1. Fault occurs in α -max:

- $|N|=24$ $k=4$ $u_{\max} = \lfloor 24/4 \rfloor = 6$

- Delete the faulty node
- Final structure:

8 8 8 (Total nodes:24)

- Using the formula, and calculating timeslots, T1 and T2:

T1:7 T2:3 T1+T2:10

```
Enter number of nodes: 25
Enter value of k: 4
T1: 7
T2: 3
T1+T2: 10
Number of alpha ring(s): 3
Alpha rings nodes: [9, 8, 8]

Single node fault occurs!

T1: 7
T2: 3
T1+T2: 10
In which alpha ring does fault occur?
Enter 1 for Max Alpha ring:
Enter 2 for Min Alpha ring:
1
New list of alpha rings with nodes in each ring:

[8, 8, 8]
```

2. Fault occurs in α -min:

- $|N|=24$ $k=4$ $u_{\max} = \lfloor 24/4 \rfloor=6$
 - Delete the faulty node from α -min ring
 - newNode= alphaMax.last_node
 - Insert newNode in alphaMin

- Delete alphaMax.last_node

- Final structure:

8 8 8 (Total nodes:24)

- Using the formula, and calculating timeslots, T1 and T2:

T1:7 T2:3 T1+T2:10

```
Enter number of nodes: 25
Enter value of k: 4
T1: 7
T2: 3
T1+T2: 10
Number of alpha ring(s): 3
Alpha rings nodes: [9, 8, 8]

Single node fault occurs!

T1: 7
T2: 3
T1+T2: 10
In which alpha ring does fault occur?
Enter 1 for Max Alpha ring:
Enter 2 for Min Alpha ring:
2
1.Delete last node from alpha min
2.newNode=alphaMax.last_node
3.Insert newNode in alphaMin
4.Delete alphaMax.last_node
New list of alpha rings with nodes in each ring:

[8, 8, 8]
```

Example-2:

- $|N|=29$ $k=4$ $u_{\max} = \lfloor 29/4 \rfloor = 7$

- Since u_{\max} is odd, multiple rings will be formed.
- Number of β -rings: 1
- Number of α -rings: $(u_{\max} - 3/2) = 2$
- β -ring will be assigned with $3k$ nodes, i.e 12
- Each α -ring will be assigned with $2k$ nodes:

12 8 8 (Total nodes:28)

- This value only adds up to 28, the remaining node will be assigned to the β - ring till $N_{\beta} \leq (2T_1 + 1)$:

13 8 8 (Total nodes:29)

- This structure contains one β -ring and two α -rings.
- Using the formula, and calculating timeslots, T_1 and T_2 :

$T_1:6$ $T_2:3$ $T_1+T_2:9$

Single node fault occurs:

Cases:

1. Fault occurs in beta-ring:

- $|N|=28$ $k=4$ $u_{\max} = \lfloor 28/4 \rfloor = 7$

- Delete the faulty node from β -ring
- Final structure:

12 8 8 (Total nodes:28)

- Using the formula, and calculating timeslots, T1 and T2:

T1:6 T2:3 T1+T2:9

```

Enter number of nodes: 29
Enter value of k: 4
T1: 6
T2: 3
T1+T2: 9
Number of alpha ring(s): 2
Number of beta ring(s): 1
Alpha rings nodes: [8, 8]
Beta ring nodes: 13

Single node fault occurs!

T1: 6
T2: 3
T1+T2: 9
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
1
1.Delete a node from beta ring structure
Nodes in beta ring: 12
Alpha rings with nodes in each ring: [8, 8]

```

2. Fault occurs in α -ring:

- $|N|=28$ $k=4$ $u_{\max} = \lfloor 28/4 \rfloor = 7$
 - Delete the faulty node from α - ring
 - Since ring now contains less than $2k$ nodes which is not acceptable for an α -ring, we shift one node from β -node to this α -node
 - $\text{newNode} = \text{betaRing.last_node}$
 - Insert newNode in alphaRing
 - Delete $\text{betaRing.last_node}$

- Final structure:

12 **8** **8** (Total nodes:28)

- Using the formula, and calculating timeslots, T1 and T2:

T1:6 T2:3 T1+T2:9


```

Enter number of nodes: 29
Enter value of k: 4
T1: 6
T2: 3
T1+T2: 9
Number of alpha ring(s): 2
Number of beta ring(s): 1
Alpha rings nodes: [8, 8]
Beta ring nodes: 13

Single node fault occurs!

T1: 6
T2: 3
T1+T2: 9
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
2
1.Delete a node from beta ring structure
2.newNode=betaRing.last_node
3.Insert newNode in alpha ring
4.Delete betaRing.last_node
Nodes in beta ring: 12
Alpha rings with nodes in each ring: [8, 8]

```

Example-3:

- $|N|=30$ $k=4$ $u_{\max} = \lfloor 30/4 \rfloor = 7$
- Since u_{\max} is odd, multiple rings will be formed.
- Number of β -rings: 1
- Number of α -rings: $(u_{\max} - 3/2) = 2$
- B-ring will be assigned with $3k$ nodes, i.e 12
- Each α -ring will be assigned with $2k$ nodes:

12 8 8 (Total nodes:28)

- This value only adds up to 28, the remaining node will be assigned to the β - ring till $N_{\beta} \leq (2T1+1)$:

13 8 8 (Total nodes:29)

Still, it is one node short, which will be assigned to one of the α -rings

13 9 8 (Total nodes:30)

- This structure contains one β -ring , one α -max and α -min rings
- Using the formula, and calculating timeslots, T1 and T2:

T1:6 T2:3 T1+T2:9

Single node fault occurs:

Cases:

1. Fault occurs in beta-ring:

- $|N|=29$ $k=4$ $u_{\max} = \lfloor 29/4 \rfloor = 7$
- Delete the faulty node from beta ring
- Final structure:
 - **12 9 8 (Total nodes:29)**
 - Using the formula, and calculating timeslots, T1 and T2:
T1:6 T2:3 T1+T2:9

```

Enter number of nodes: 30
Enter value of k: 4
T1: 6
T2: 3
T1+T2: 9

Number of alpha ring(s): 2
Number of beta ring(s): 1
Alpha rings nodes: [9, 8]
Beta ring nodes: 13

Single node fault occurs!

T1: 6
T2: 3
T1+T2: 9
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
1
1.Delete a node from beta ring structure
Nodes in beta ring: 12
Alpha rings with nodes in each ring: [9, 8]

```

2. Fault occurs in α -max ring:

- $|N|=29$ $k=4$ $u_{max} = \lfloor 29/4 \rfloor=7$
- Delete the faulty node from α -max ring

- Final structure:

| | | | |
|-----------|----------|----------|------------------|
| 13 | 8 | 8 | (Total nodes:29) |
|-----------|----------|----------|------------------|

- Using the formula, and calculating timeslots, T1 and T2:

| | | |
|------|------|---------|
| T1:6 | T2:3 | T1+T2:9 |
|------|------|---------|

```

Enter number of nodes: 30
Enter value of k: 4
T1: 6
T2: 3
T1+T2: 9

Number of alpha ring(s): 2
Number of beta ring(s): 1
Alpha rings nodes: [9, 8]
Beta ring nodes: 13

Single node fault occurs!

T1: 6
T2: 3
T1+T2: 9
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
2
In which alpha ring does fault occur?
Enter 1 for MaxAlpha ring:
Enter 2 for MinAlpha ring:
1
Nodes in beta ring: 13
Alpha rings with nodes in each ring: [8, 8]

```

3. Fault occurs in α -min ring:

- $|N|=29$ $k=4$ $u_{\max} = \lfloor 29/4 \rfloor = 7$
 - Delete the faulty node from α -min ring
 - Since node now contains less than $2k$ nodes which is not acceptable for an α -ring, we shift one node from β -node to this α -node
 - $\text{newNode} = \text{alphaMax.last_node}$ since α -max ring contains more than $2k$ nodes
 - Insert newNode in α Min ring
 - Delete $\text{alphaMax.last_node}$

- Final structure:

13 8 8 (Total nodes:29)

- Using the formula, and calculating timeslots, T1 and T2:

T1:6 T2:3 T1+T2:9

```
Enter number of nodes: 30
Enter value of k: 4
T1: 6
T2: 3
T1+T2: 9
.
Number of alpha ring(s): 2
Number of beta ring(s): 1
Alpha rings nodes: [9, 8]
Beta ring nodes: 13

Single node fault occurs!

T1: 6
T2: 3
T1+T2: 9
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
2
In which alpha ring does fault occur?
Enter 1 for MaxAlpha ring:
Enter 2 for MinAlpha ring:
2
Delete last node from alpha min
newNode=alphaMax.last_node
Insert newNode in alphaMin
Delete alphaMax.last_node
Nodes in beta ring: 13
Alpha rings with nodes in each ring: [8, 8]
```

Example-4:

- $|N|=28$ $k=3$ $u_{\max} = \lfloor 28/3 \rfloor = 9$

- Since u_{\max} is odd, multiple rings will be formed.

- Number of β -rings: 1

- Number of α -rings: $(u_{\max} - 3/2) = 3$

- β -ring will be assigned with $3k$ nodes, i.e 9

Each α ring will be assigned with $2k$ nodes:

9 6 6 6 (Total nodes:27)

- This value only adds up to 27, the remaining node will be assigned to the β -ring till $N_{\beta} \leq (2T1+1)$ i.e 9, which cannot happen in this case

So, it is one node short, which will be assigned to one of the α rings

9 **7** **6** **6** (Total nodes:28)

- This structure contains one beta-ring , one α -max and two α -min rings

- Using the formula, and calculating timeslots, T1 and T2:

T1:4 T2:4 T1+T2:8

Single node fault occurs:

Cases:

1. Fault occurs in beta-ring:

- $|N|=27$ $k=3$ $u_{\max} = \lfloor 27/3 \rfloor = 9$

- Delete the faulty node from β -ring

- Since nodes in β -ring become less than $3k$, which is the minimum requirement, we shift a node from α -max ring to β -ring
 - $\text{newNode} = \text{alphaMax.last_node}$ since α -max ring contains more than $2k$ nodes
 - Insert newNode in β -ring
 - Delete $\text{alphaMax.last_node}$

- Final structure:

9 6 6 6 (Total nodes:27)

- Using the formula, and calculating timeslots, T1 and T2:

T1:4 T2:4 T1+T2:8

```

Enter number of nodes: 28
Enter value of k: 3
T1: 4
T2: 4
T1+T2: 8

Number of alpha ring(s): 3
Number of beta ring(s): 1
Alpha rings nodes: [7, 6, 6]
Beta ring nodes: 9

Single node fault occurs!

T1: 4
T2: 4
T1+T2: 8
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
1
1.Delete a node from beta ring structure
2.newNode=alphaMax.last_node
3.Insert newNode in beta ring
4.Delete alphaMax.last_node
Nodes in beta ring: 9
Alpha rings with nodes in each ring: [6, 6, 6]

```

2. Fault occurs in α -max ring:

- $|N|=27$ $k=3$ $u_{\max} = \lfloor 27/3 \rfloor = 9$
 - Delete the faulty node from α -max ring
 - Final structure:
- 9** **6** **6** **6** (Total nodes:27)

- Using the formula, and calculating timeslots, T1 and T2:

T1:4 T2:4 T1+T2:8

```
Enter number of nodes: 28
Enter value of k: 3
T1: 4
T2: 4
T1+T2: 8

Number of alpha ring(s): 3
Number of beta ring(s): 1
Alpha rings nodes: [7, 6, 6]
Beta ring nodes: 9

Single node fault occurs!

T1: 4
T2: 4
T1+T2: 8
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
2
In which alpha ring does fault occur?
Enter 1 for MaxAlpha ring:
Enter 2 for MinAlpha ring:
1
Nodes in beta ring: 9
Alpha rings with nodes in each ring: [6, 6, 6]
```

3. Fault occurs in α -min ring:

- $|N|=27$ $k=3$ $u_{\max} = \lfloor 27/3 \rfloor=9$
 - Delete the faulty node from α -min ring
 - Since node now contains less than $2k$ nodes which is not acceptable for an α -ring, we shift one node from α -max to α -min ring (Also, since the difference in nodes between any two α -rings cannot be more than 1)
 - $\text{newNode} = \text{alphaMax.last_node}$

(since α -max ring contains more than $2k$ nodes)
 - Insert newNode in alphaMin ring
 - Delete $\text{alphaMax.last_node}$

- Final structure:

9 6 6 6 (Total nodes:27)

- Using the formula, and calculating timeslots, T1 and T2:

T1:4 T2:4 T1+T2:8

```

Enter number of nodes: 28
Enter value of k: 3
T1: 4
T2: 4
T1+T2: 8

Number of alpha ring(s): 3
Number of beta ring(s): 1
Alpha rings nodes: [7, 6, 6]
Beta ring nodes: 9

Single node fault occurs!

T1: 4
T2: 4
T1+T2: 8
In which ring does fault occur?
Enter 1 beta ring:
Enter 2 for alpha ring:
2
In which alpha ring does fault occur?
Enter 1 for MaxAlpha ring:
Enter 2 for MinAlpha ring:
2
Delete last node from alpha min
newNode=alphaMax.last_node
Insert newNode in alphaMin
Delete alphaMax.last_node
Nodes in beta ring: 9
Alpha rings with nodes in each ring: [6, 6, 6]

```

❖ Cases when u_{\max} changes: Structure and number of rings also change

Example-5:

- $|N|=27$ $k=3$ $u_{\max} = \lfloor 27/3 \rfloor=9$
- Since u_{\max} is odd, multiple rings will be formed.
- Number of β -rings: 1

- Number of α rings: $(u_{\max}-3/2)=3$
- β -ring will be assigned with $3k$ nodes, i.e 9
- Each α -ring will be assigned with $2k$ nodes:

9 6 6 6 (Total nodes:27)

- This structure contains one β -ring , three α -rings
- Using the formula, and calculating timeslots, T1 and T2:

T1:4 T2:3 T1+T2:7

Single node fault occurs:

- $|N|=26$ $k=3$ $u_{\max} = \lfloor 26/3 \rfloor=8$
- u_{\max} changes to even value from odd value
- Since u_{\max} is even, only α -rings will be formed
- Therefore, we need to reconstruct α -rings using $|N|-1$ nodes
- Number of α rings: $(u_{\max} /2)=4$
- Each α -ring will be assigned with $2k$ nodes:

6 6 6 6 (Total nodes:24)

- Remaining nodes will be assigned to α -rings one by one

7 7 6 6 (Total nodes:26)

Structure contains two α -max rings and two α -min rings

- Using the formula, and calculating timeslots, T1 and T2:

T1:5 T2:3 T1+T2:8

```

Enter number of nodes: 27
Enter value of k: 3
T1: 4
T2: 3
T1+T2: 7
Number of alpha ring(s): 3
Number of beta ring(s): 1
Alpha rings nodes: [6, 6, 6]
Beta ring nodes: 9

Single node fault occurs!

T1: 5
T2: 3
T1+T2: 8
Number of alpha ring(s): 4
Alpha rings nodes: [7, 7, 6, 6]

```

Example-6:

- $|N|=24$ $k=4$ $u_{\max} = \lfloor 24/4 \rfloor=6$

- Since u_{\max} is even, only α -rings will be formed
- Number of α rings: $(u_{\max} / 2)=3$

Each α -ring will be assigned with $2k$ nodes:

8 8 8 (Total nodes:24)

- This structure contains three α -rings
- Using the formula, and calculating timeslots, T1 and T2:

T1:7 T2:2 T1+T2:9

Single node fault occurs:

- $|N|=23$ $k=4$ $u_{\max} = \lfloor 23/4 \rfloor = 5$
- u_{\max} changes to odd value from even value
- Since u_{\max} is odd, multiple rings will be formed
- Therefore, we need to reconstruct these rings using $|N|-1$ nodes
- Number of β -rings: 1
- Number of α -rings: $(u_{\max}-3/2)=1$
- β -ring will be assigned with $3k$ nodes
- α -ring will be assigned with $2k$ nodes:

12 8 (Total nodes:20)

- This value only adds up to 20, the remaining node will be assigned to the β -ring till $N_{\beta} \leq (2T1+1)$ i.e 15,

15 8 (Total nodes:23)

Structure contains one β -ring and α -ring each

- Using the formula, and calculating timeslots, T1 and T2:

T1:7 T2:2 T1+T2:9

```
Enter number of nodes: 24
Enter value of k: 4
T1: 7
T2: 2
T1+T2: 9
Number of alpha ring(s): 3
Alpha rings nodes: [8, 8, 8]

Single node fault occurs!

T1: 7
T2: 2
T1+T2: 9
Number of alpha ring(s): 1
Number of beta ring(s): 1
Alpha rings nodes: [8]
Beta ring nodes: 15
```

4.3 Results and data set used for analysis of the algorithm:

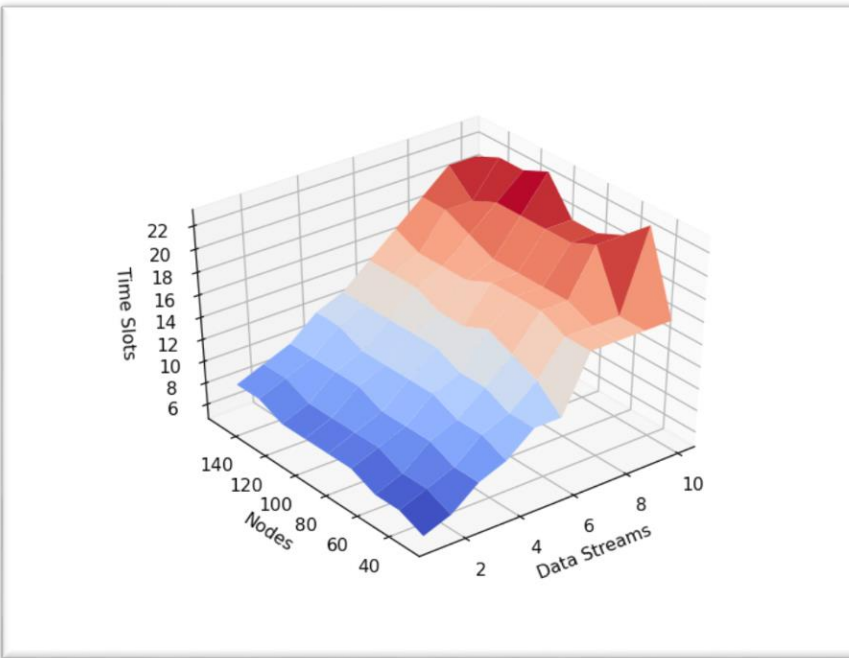
For the data set, the total nodes (N) considered are 150 and total number of data streams (k) are 10.

There are two nested loops, the outer loop considers values (representing nodes) starting from 30 to 150 in intervals of 15 while the inner loop considers all values (representing data streams) from 1 to 10. Time-slots are calculated for each of the cases using `timeslots_actual()` function, and also their structures are defined through the `construct()` function. Then, for each of these cases, we consider the occurrence of fault in a single node. Timeslots for (N-1) nodes and k data streams are again calculated using `timeslots_error()` and new structures are defined depending upon if change occurs in u_{\max} or not.

From the following set of data we can observe how the 3D surface plot structure changes with subject to fault in a node. Fig 12 shows the time-slots when there is no faulty node in the network while Fig 13 shows the time-slots after a node becomes faulty. There is only a slight difference in the 3D plot structure, which is visible at the top region of the plot and especially at the top-right corner. In conclusion, only a slight difference in the values of time-slots is shown by these 3D plots. Even with the occurrence of a faulty node it has a low impact on the number of time-slots and eventually on the performance

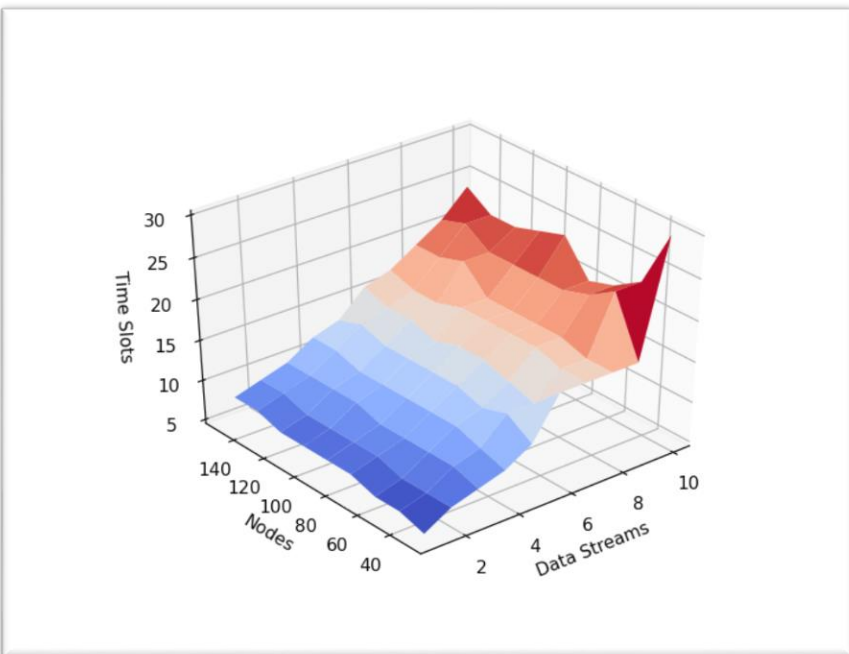
RESULTS:

Fig 12. Before fault occurs:



Nodes = 150
Data Streams = 10
Faulty node = 0

Fig 13. After fault occurs:



Nodes = 150
Data Streams = 10
Faulty node = 1

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

With consistently expanding interest of IoT devices it gets hard to keep up a system with minimum delay and in certifiable fault free. Therefore in this undertaking I have effectively conceived an algorithm for concurrent data collection trees and it's implementation that guarantee information collection time with least delay. This algorithm directs the issue of a faulty node. Also an algorithm was developed for fault tolerance(where a single node becomes faulty) and mechanism was developed through which we can handle the fault without much overhead. The algorithm has also been implemented in Python which gives the new structures of the rings after a fault has occurred.

LIST OF REFERENCES

- [1] C-T. Cheng, N. Ganganath, and K. Fok, "Concurrent data collection trees for IoT applications, " *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 793–799, September 2017.
- [2] C.-T. Cheng, C. K. Tse, and F. C. M. Lau, "A delay-aware data collection network structure for wireless sensor networks," *IEEE Sensors J.*, vol. 11, no. 3, pp. 699–710, Mar. 2011.
- [3] X. M. Huang, J. Deng. J. Ma, and Z. Wu;, "Fault tolerant routing for wireless sensor grid networks," *In Proceedings of the IEEE Sensors Applications Symposium*, pp. 66-70, 2006.
- [4] G. Alari, J. Beauquier, J. Chacko, A.K. Datta, S. Tixeuil, A fault-tolerant distributed sorting algorithm in tree networks, 26 A. Sasaki / *Information Processing Letters* 83 (2002) 21–26 in: Proc. 1998 IEEE Internet. Performance, Computing and Communications Conf., 1998, pp. 37 -43.
- [5] Mihaela Cardei , Shuhui Yang , Jie Wu, Algorithms for Fault-Tolerant Topology in Heterogeneous Wireless Sensor Networks, *IEEE Transactions on Parallel and Distributed Systems*, v.19 n.4, p.545-558, April 2008
- [6] Y. Zeng, L. Xu, Z. Chen, "Fault-tolerant algorithms for connectivity restoration in wireless sensor networks", *Sensors*, vol. 16, no. 1, pp. 3, 2016.
- [7] K. Rajeswari and S. Neduncheliyan, "Genetic algorithm based fault tolerant clustering in wireless sensor network," *IET Communications*, vol. 11, no. 12, pp. 1927–1932, 2017.
- [8] Elmira Moghaddami Khalilzad, Sanam Hosseini, "Recovery of Faulty Cluster Head Sensors by Using Genetic Algorithm (RFGA) ", *International Journal of Computer Science Issues*, Vol.9, No. 4, pp. 141-145, 2012.
- [9] Ghaffari, A., & Nobahary, S. (2015). FDMG: Fault detection method by using genetic algorithm in clustered wireless sensor networks. *Journal of AI and Data Mining*, 3(1), 47–57.

APPENDIX - A

I) Case A - When Node becomes faulty

$$u_{max} = \lfloor \frac{|N| - \hat{n}}{k} \rfloor$$

Proof:

$$\text{for } \hat{n} = 1 \quad u_{max} = \lfloor \frac{|N|-1}{k} \rfloor$$

$$\text{for } \hat{n} = 2 \quad u_{max} = \lfloor \frac{|N|-2}{k} \rfloor$$

$$\text{for } \hat{n} = n \quad u_{max} = \lfloor \frac{|N|-n}{k} \rfloor$$

for \hat{n} nodes

Where u_{max} = maximum number of nodes that can communicate

N : total number of nodes in the network

k : total number of data streams or base stations

$$u_{max} = \lfloor \frac{|N| - \hat{n}}{k} \rfloor$$

161338

ORIGINALITY REPORT

18%

SIMILARITY INDEX

13%

INTERNET SOURCES

14%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

| | | |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1 | Chi-Tsun Cheng, Nuwan Ganganath, Kai-Yin Fok. "Concurrent Data Collection Trees for IoT Applications", IEEE Transactions on Industrial Informatics, 2017 Publication | 5% |
| 2 | www.ir.juit.ac.in:8080 Internet Source | 4% |
| 3 | Mihaela Cardei. "Algorithms for Fault-Tolerant Topology in Heterogeneous Wireless Sensor Networks", IEEE Transactions on Parallel and Distributed Systems, 04/2008 Publication | 3% |
| 4 | ira.lib.polyu.edu.hk Internet Source | 2% |
| 5 | www.cse.fau.edu Internet Source | 1% |
| 6 | researchbank.rmit.edu.au Internet Source | 1% |

| | | |
|----|------------------------------------------------------------------------------------------|-----|
| | Internet Source | <1% |
| 8 | repository.up.ac.za Internet Source | <1% |
| 9 | www.slideshare.net Internet Source | <1% |
| 10 | www.scribd.com Internet Source | <1% |
| 11 | citeseerx.ist.psu.edu Internet Source | <1% |
| 12 | dschleef.umwblogs.org Internet Source | <1% |
| 13 | ir.uitm.edu.my | <1% |

Internet Source

14

Sachi Nandan Mohanty, E. Laxmi Lydia, Mohamed Elhoseny, Majid M. Gethami Al Otaibi, K. Shankar. "Deep learning with LSTM based distributed data mining model for energy efficient wireless sensor networks", Physical Communication, 2020

$< 1\%$

Publication

15

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.

$< 1\%$

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 15/07/2020

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

Name: Hima K S Department: CSE Enrolment No 161338

Contact No. 9013082185 E-mail. himaks90@gmail.com

Name of the Supervisor: Mr. Arvind Kumar

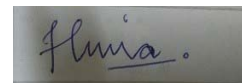
Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): Fault Tolerance in IoT

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 = 61
- Total No. of Preliminary pages = 08
- Total No. of pages accommodate bibliography/references = 02



(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 18 %. 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) | |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------|----------------------|------------------------------------------------------------------|--|
| | <ul style="list-style-type: none">• 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