FAULT TOLERANCE IN IOT

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

In

Computer Science and Engineering

By

Aruba Sood (161245)

Under the supervision of

Mr. Arvind Kumar

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

CERTIFICATE

I hereby declare that the work presented in this report entitled **"Fault Tolerance in IoT "** in partial fulfillment of the requirements 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 May 2020 under the supervision of **Mr. Arvind Kumar** (Assistant Professor (Grade-II) Department of Computer Science Engineering & IT).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Aruba Sood
161245

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

Mr. Arvind Kumar

Assistant Professor (Grade-II)

Department of Computer Science Engineering & IT

Dated:

ACKNOWLEDGEMENT

First of all, I would like to express our 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. He has always been my motivation for carrying out the project. His constant encouragement at every step was a precious asset during my project work.

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

I am thankful to **Mr. Ravi Raina** for providing endless support and entertaining all my problems. Also I am thankful to the faculty and staff of Department of Computer Science Engineering & IT, Jaypee University of Information Technology for providing me all the facilities required for experimental work.

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

Aruba Sood

TABLE OF CONTENTS

LIST OF ACRONYMS AND ABBREVIATIONS

IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
WSN	Wireless Sensor Network
DADCNS	Delay-Aware Data Collection Network Structure
MST	Minimum Spanning Tree
MAC	Media Access Control
CDMA	Code Division Multiple Access
DCT	Data Collection Time
N2N	Node to Node
N2BS	Node to Base Station

LIST OF FIGURES

ABSTRACT

IOT nodes are inclined to failure in hostile situations subsequently, in this paper, a fault tolerant algorithm is portrayed for a IOT network - Concurrent Data Collection Trees. As the system becomes broken it ought to be reestablished to its typical working inside a effective time period. 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. We developed an algorithm for fault tolerance in both topologies.

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 we are going to develop in Java.

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 = {n1, n2, n3, ……,n|N|} nodes and S = {s1, s2, s3, ….., 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|≥k

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

u_{max} = floor(|N| / k) (1)

u_i = min[u_{max} , |N| − $\sum i-1$ $\hat{u_j}$] (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 $|N| - \tau 1\lceil u_{max} \rceil$.

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

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

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 $umax = 2$. "A data stream in a α-ring Nα will require T1 time slots to aggregate information from |Nα| 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 Ƙth data collection process) i.e. node nc1 transmits information to node nc2

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

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

Data fusion will be performed on nodes n_{c2}.Overall duration can be calculated using T_1 and T_2 from above equations.

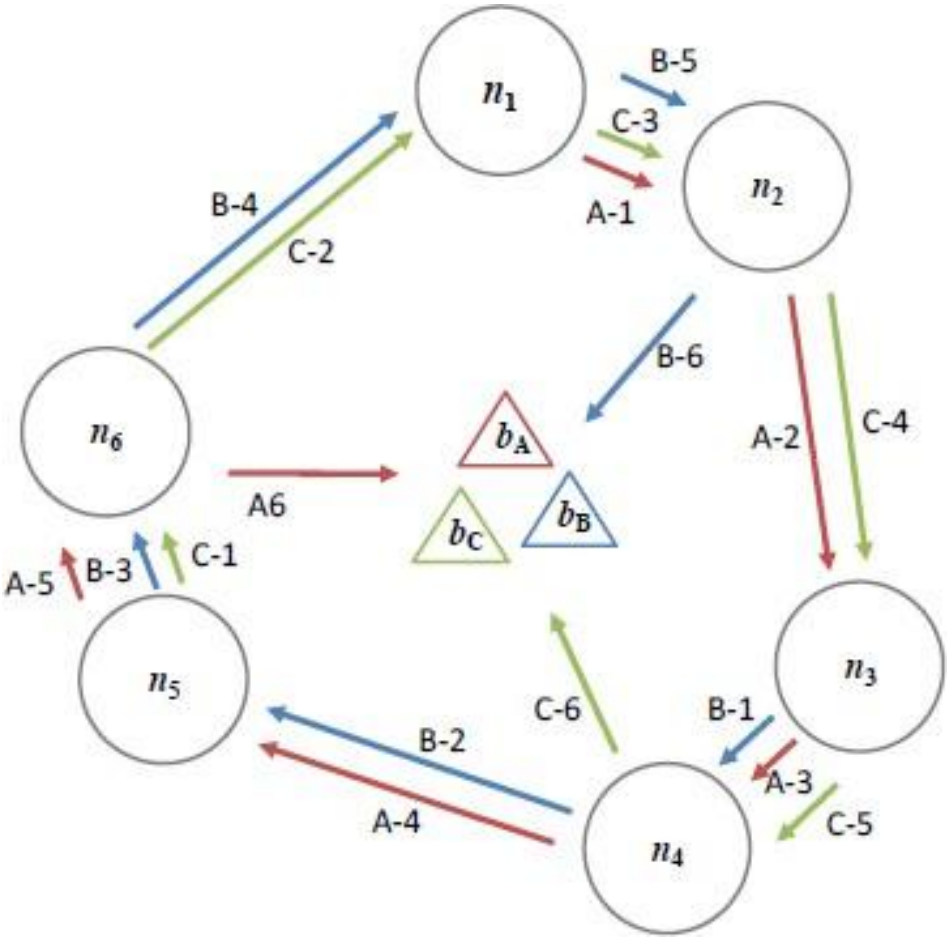

Fig 1 Alpha ring

- The circles represent the nodes
- The triangles represent the base stations.
- A particular stream of arrows represents a particular concurrent data stream.

In this case:

- We have 6 nodes with 3 base stations.
- Red arrows- A stream
- Blue arrows- B stream
- Green arrows –C stream

B) The β-ring

For $_{u\ max}$ = 3 and |Nβ| ≥ 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 Ƙth data collection process node nc3 will be engaged with node to-base station communication and nc4 will transmit data nc5.

$$c3 = (1 + \mathrm{mod}(3(Ƙ-1) + 2(t-1), |Nβ|))$$

$$c4 = (1 + \mathrm{mod}(3(Ƙ-1) + 2(t-1) + 1, |Nβ|))$$

$$c5 = (1 + \mathrm{mod}(3(Ƙ-1) + 2(t-1) + 2, |Nβ|))$$

Data fusion will be performed on node nc5.

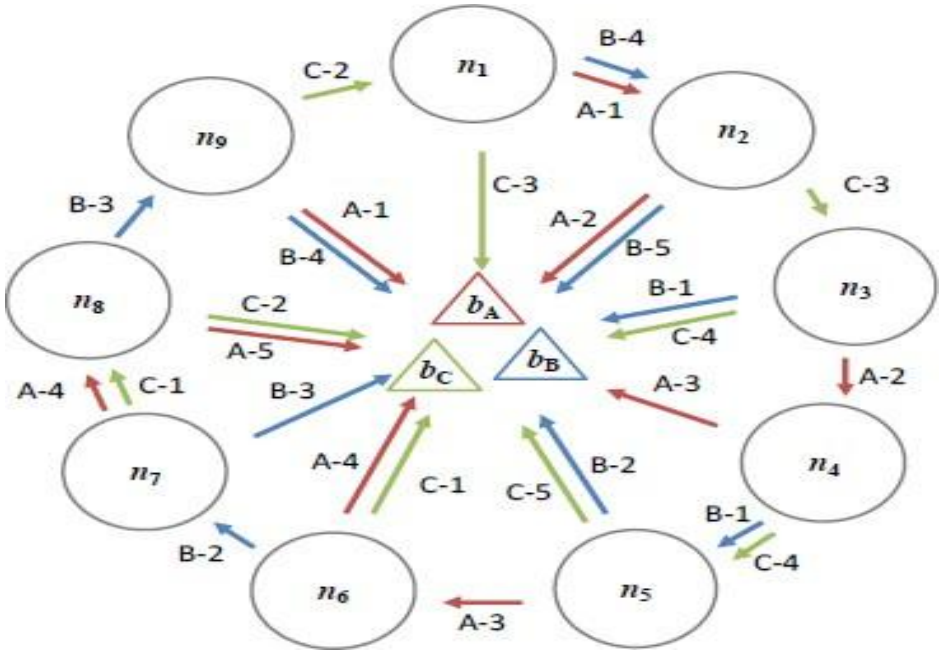

Fig. 2 Beta ring

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|-nα(2k) nodes will at that point be granted to those nα 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'α(2kBS) node will be allowed to the β-ring until |Nβ| = 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 lineraly 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 multihop 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 ðk-ATCÞ 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 for Data Collection Trees

A) Fault tolerance algorithm for concurrent data collection trees:

When single node becomes faulty :

Start

Calculate umax= ⌊ n/k ⌋

uTemp = umax

Calculate T1,T2

Calculate nAlpha, nBeta

Calculate nAlphaNodes, nBetaNodes

nAlpha_temp = nAlpha, nBeta_temp= nBeta

nAlphaNodes_temp= nAlphaNodes, nBetaNodes_temp= nBetaNodes

while (1)

{

if(N_{new} == N)

{

Calculate umax= ⌊ n-1/k ⌋

Calculate T1,T2

Calculate Nalpha, Nbeta

Calculate Nalphanodes ,Nbetanodes

If umax== temp

If umax%2==0

If fnode exits 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)

i. Check alpha rings with > 2k nodes

ii. new=alphamax.lnode

iii. insert new in beta end

iv. delete (alphamax.lnode)

Else

If(alphamax==alphamin)

Delete(fnode)

```
                    if(nAlphaNodes<2k*nAlpha)
                        new=beta.lnode
                        insert new in alphamin end
                        delete (beta.lnode)

                Else if fnode exits 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 algorithm (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. The concluding time slot scenario works like the "Base Station Faulty" circumstance.

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.

4.2 Theoretical examples with mathematical analysis

In this section we will take some theoretical examples and see how these will work with our algorithm

4.2.1 Cases where u_{max} does not change

When u_{max} doesn't change after deletion of the faulty node and there is no transition from α-ring to β-ring and vice-versa. Only re-arrangement in the structures take place.

4.2.1.1 Example-1

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

4.2.1.1 a) Structure formation

- u_{max} is even (only α- rings would be formed).
- u_{max} /2 α- rings i.e. 3 α-rings will be formed
- Each α-ring will contain minimum of 2k nodes i.e. 8 nodes

8 8 8 (8+8+8=24)

- One node is still left so it will be assigned to the first α-ring.

8 8 8
+1
9 8 8

- First α-ring becomes the alpha_max ring with maximum number of nodes i.e. 9
- Other two are alpha_min rings with 8 nodes each.
- T1=7, T2=2; T1+T2=9
- The α-ring structures will be like following

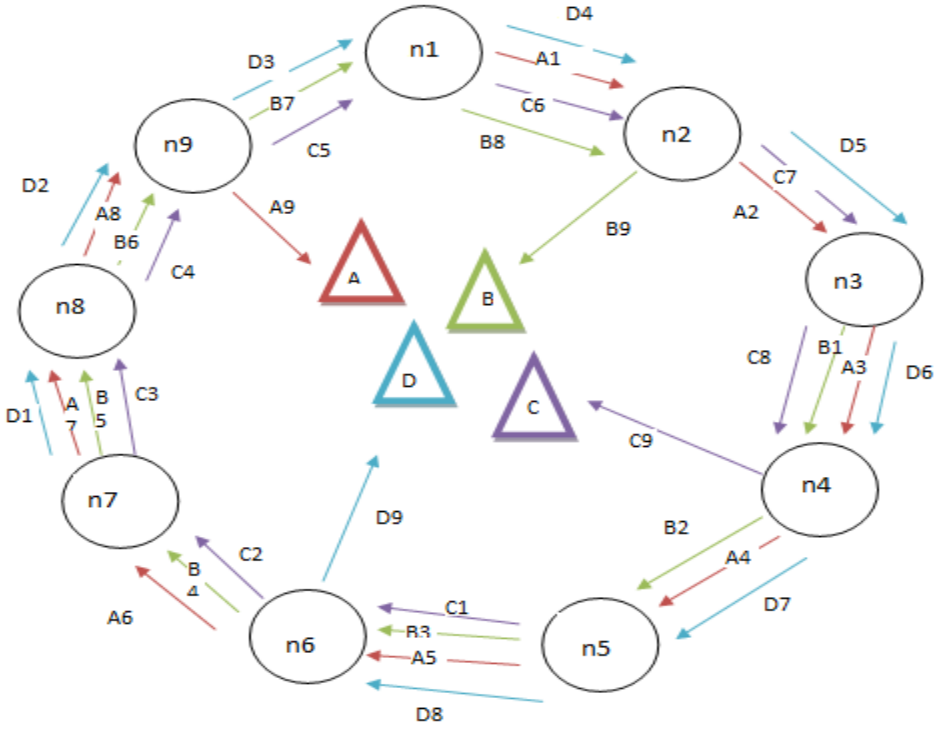

Fig. 3 First α-ring with 9 nodes (alpha_max)

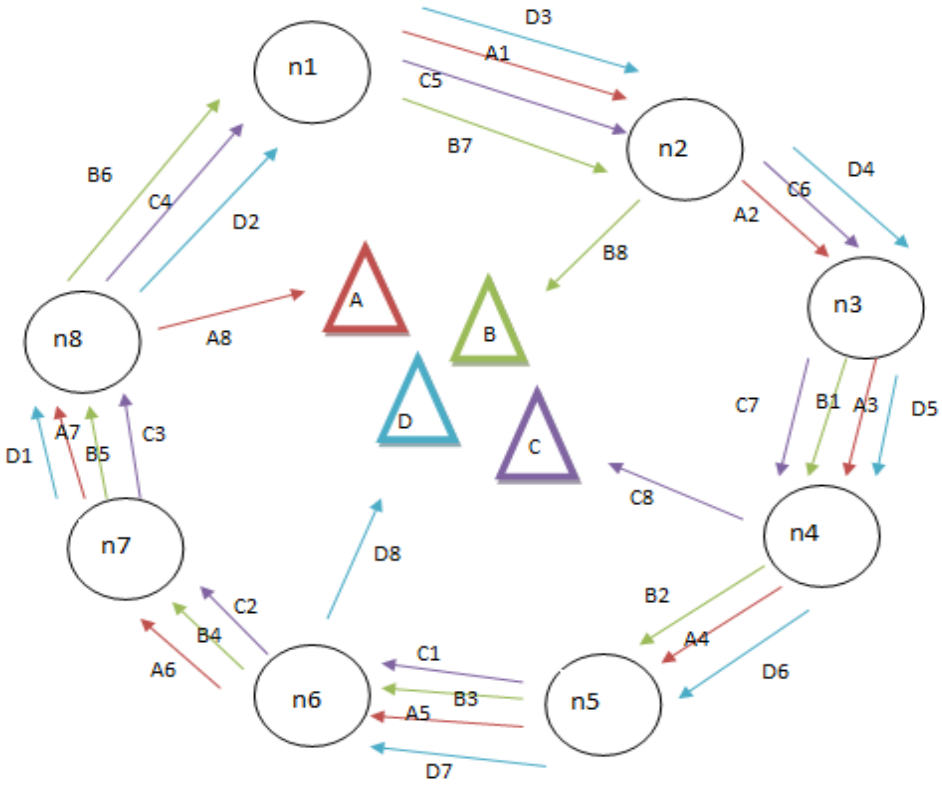

Fig. 4 2nd α-ring with 8 nodes (alpha_min)

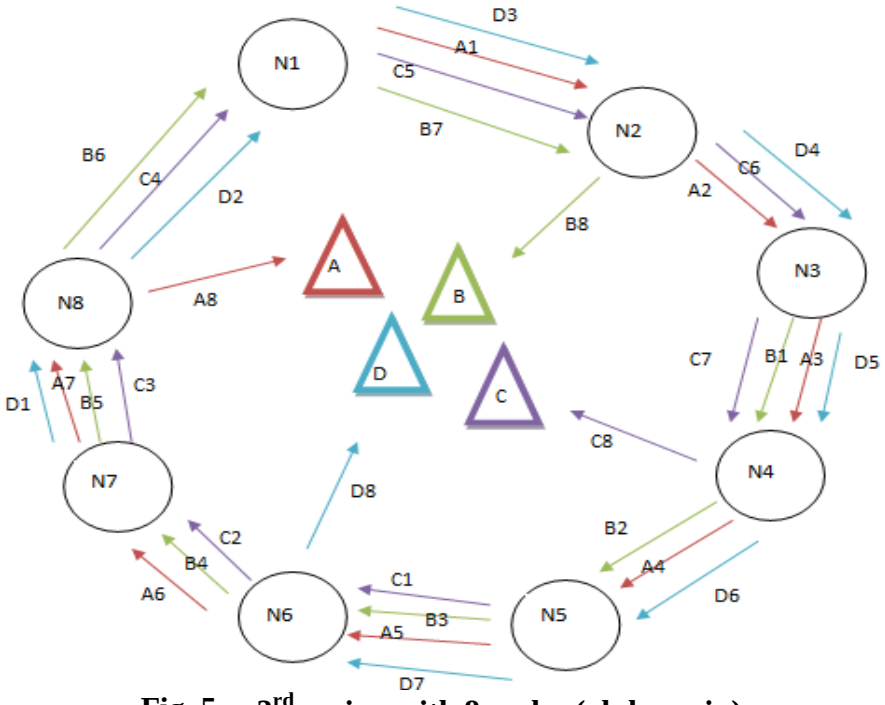

Fig. 5 3rd α-ring with 8 nodes (alpha_min)

- The tree structures of the above three rings would be as following figures

Fig. 6 First ring (apha_max) tree structure

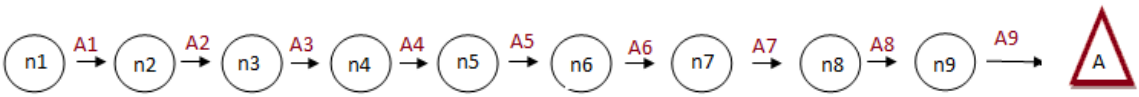

Fig. 6 a) First concurrent data stream tree

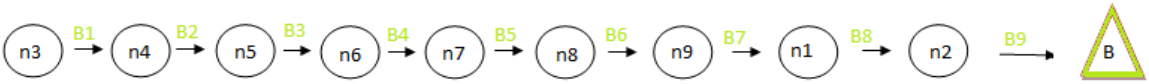

Fig. 6 b) Second concurrent data stream tree

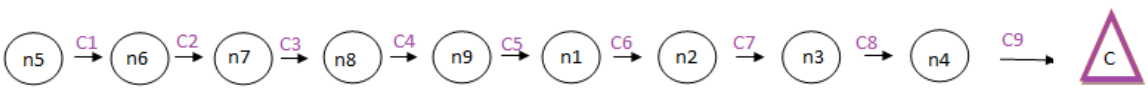

Fig. 6 c) Third concurrent data stream tree

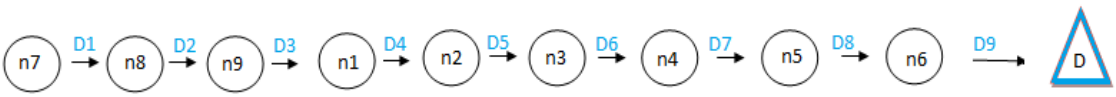

Fig. 6 d) Fourth concurrent data stream tree

Fig. 7 Second ring (alpha_min) tree structures

Fig. 7 a) First concurrent data stream tree

Fig. 7 b) Second concurrent data stream tree

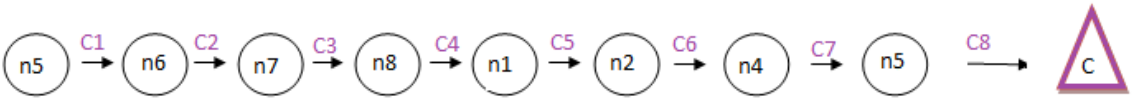

Fig. 7 c) Third concurrent data stream tree

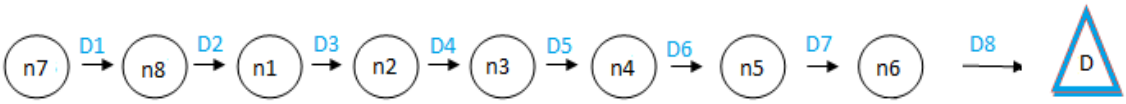

Fig. 7 d) Fourth concurrent data stream tree

- Structure of third ring α-ring is exactly same as second ring(since both contain same number of node

4.2.1.1 b) Node-Fault

Cases:

1. **If** node fault occurs in alpha_max (1st α-ring), for example –node fault occurs at node 4 of alpha_max.
    - Simply the faulty node will be deleted
    - Now the structure of all the alpha_max ring would be like following

Fig. 8 After faulty node deletion apha_max ring tree structures

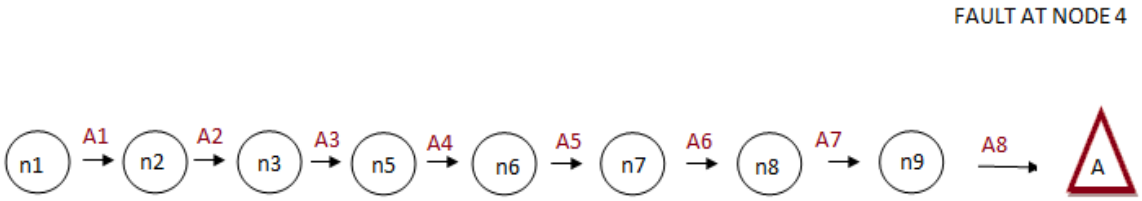

Fig. 8 a) First concurrent data stream tree

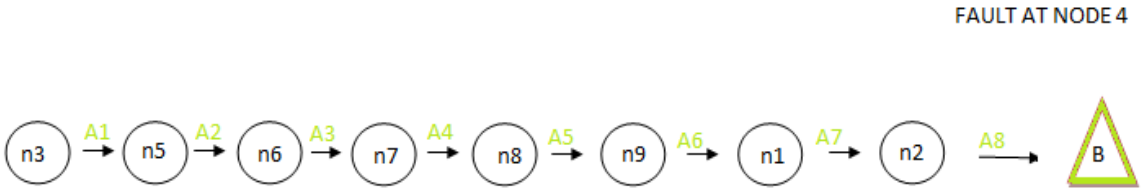

Fig. 8 b) Second concurrent data stream tree

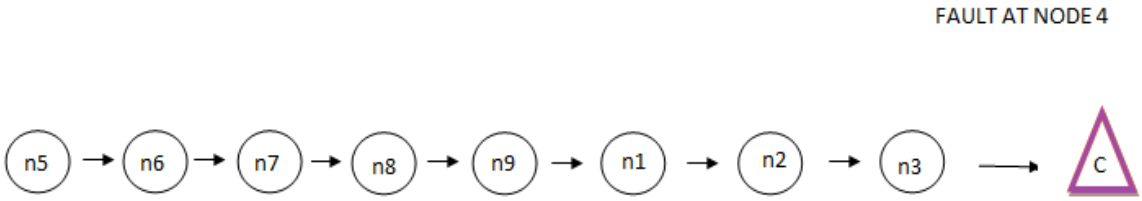

Fig. 8 c) Third concurrent data stream tree

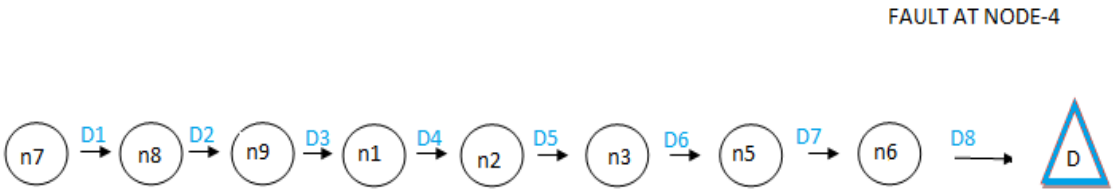

Fig. 8 d) Fourth concurrent data stream tree

- The structure of rest of the two rings will remain same.

2. **Else If** the fault occurs in one of the alpha_min rings. For example the fault occurs in 3rd ring at node 6.
   - The faulty node will be deleted.
   - The last node from the alpha_max ring will be inserted at the end of the faulty ring and deleted from the original ring.
   - The tree structure of the 3rd ring would be as following figure.

Fig. 9 After faulty node deletion 3rd ring's tree structures (apha_min)

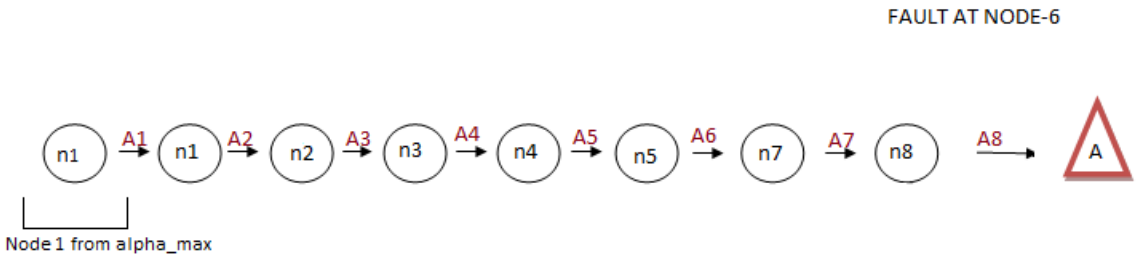

Fig. 9 a) First concurrent data stream tree

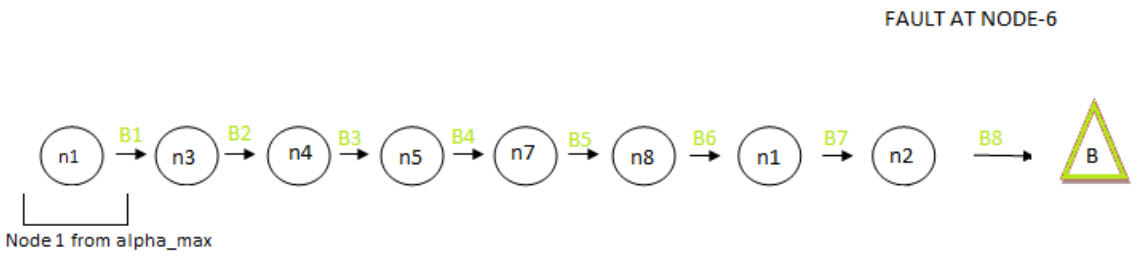

Fig. 9 b) Second concurrent data stream tree

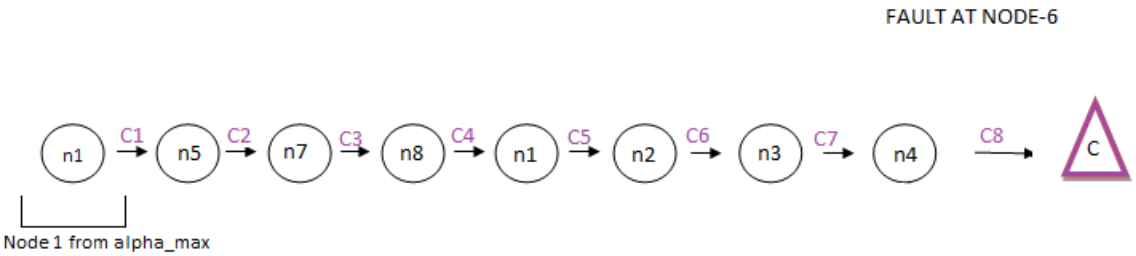

Fig. 9 c) Third concurrent data stream tree

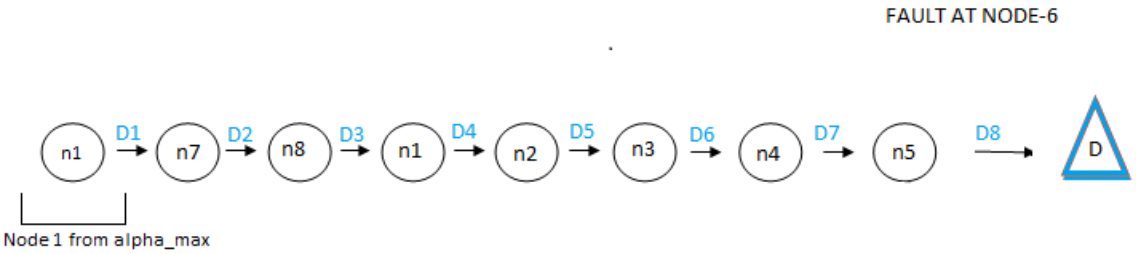

Fig. 9 d) Fourth concurrent data stream tree

- The tree structure of the 1st ring (apha_max) would be as following figure.

Fig. 10 After faulty node deletion tree structures of the alpha_max ring

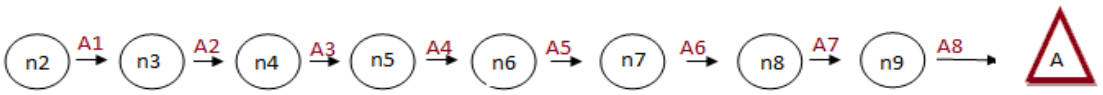

Fig. 10 a) First concurrent data stream tree

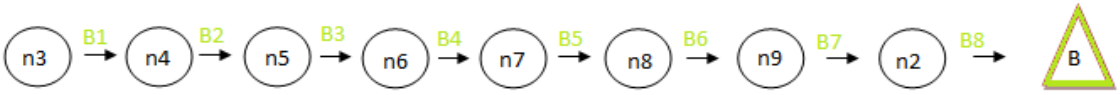

Fig. 10 b) Second concurrent data stream tree

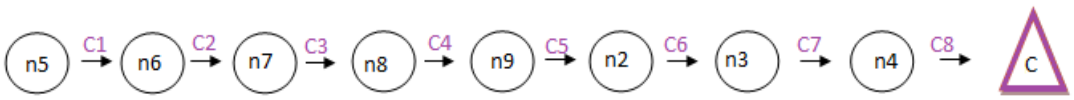

Fig. 10 c) Third concurrent data stream tree

Fig. 10 d) Fourth concurrent data stream tree

- The structure of the 3rd α-ring would not be changed

4.2.1.2 Example 2

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

4.2.1.2 a) Structure formation

- u_{max} is odd (multiple rings (both α andβ rings would be formed).
- 1β-ring with be formed.
- The β-ring with 3k nodes will be formed i.e. 12 nodes
- (umax-3)/2 α- rings i.e. 2 α-rings will be formed.
- Each α-ring will contain minimum of 2k nodes i.e. 8 nodes

  12 8 8 (12+8+8=28)
- One node is still left so it will be assigned to the β-ring until $N_\beta=2T_1+1$.

  12 8 8

  +1

  13 8 8
- β-ring contains 13 nodes
- Other two α-rings contain 8 nodes each.
- T1=6, T2=3; T1+T2=9.
- The ring structures will be as follows:

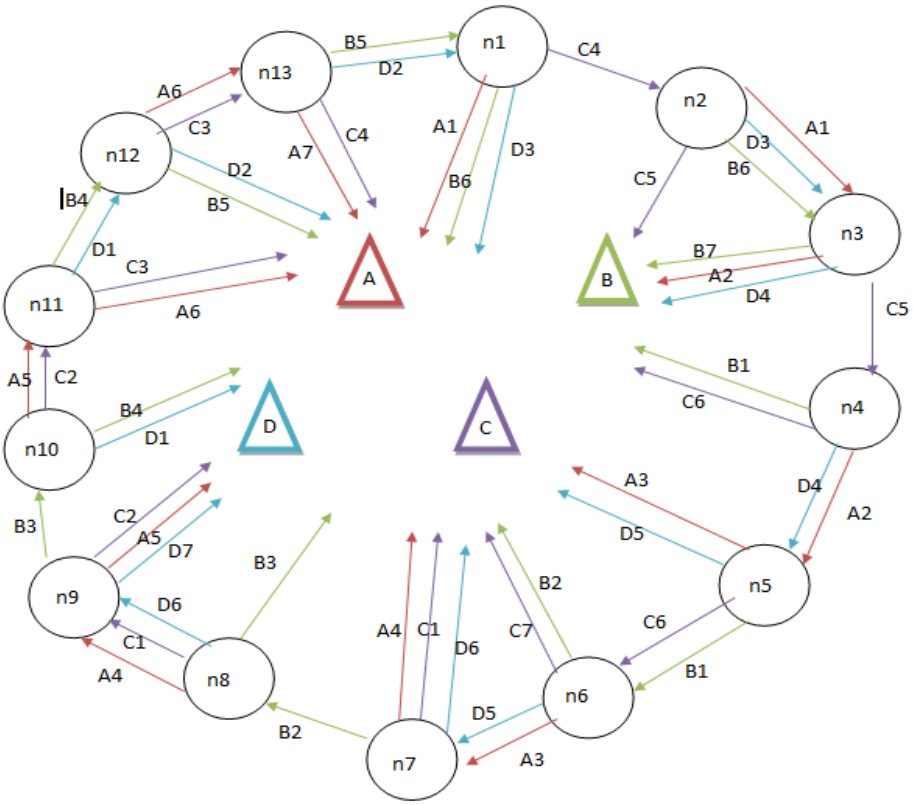

Fig. 11 **β-ring structure with 13 node**

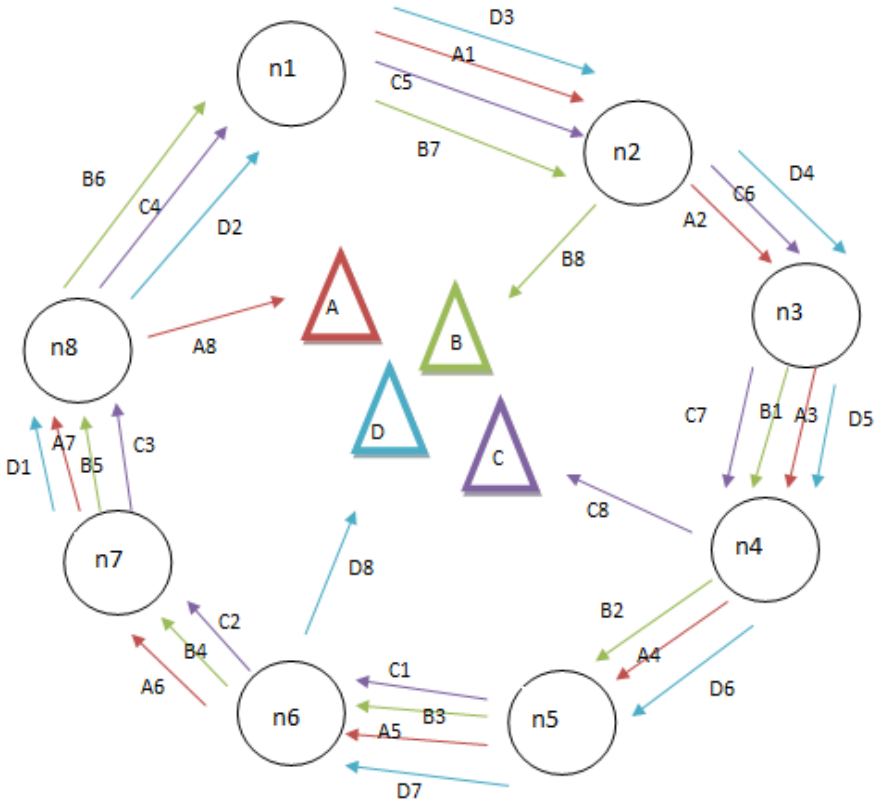

Fig. 12 **α-ring structure with 8 nodes**

- Both the 2nd and 3rd rings (α-rings) have same number of nodes. Therefore, the ring structure of both the rings would be same.
- The tree structures that can be formed from the ring structures of the above are as follows

Fig. 13 Tree structures of the β-ring(1st ring)

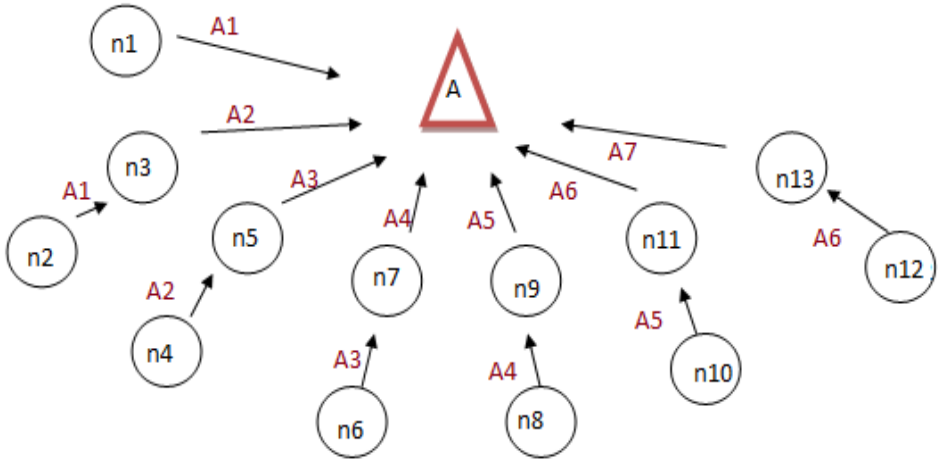

Fig. 13 a) First concurrent data stream tree

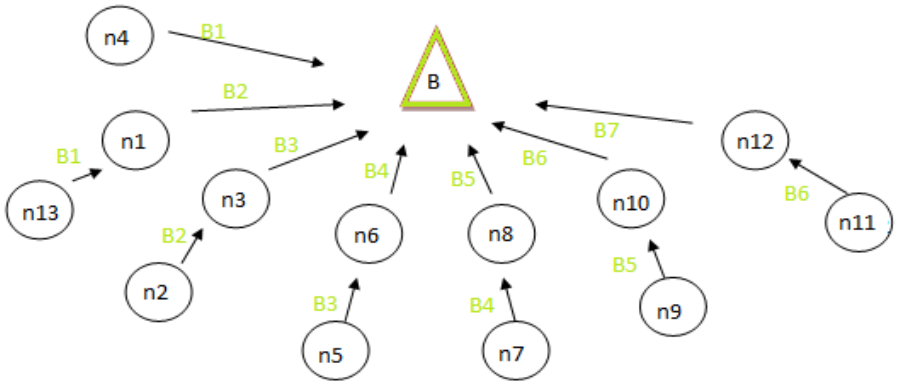

Fig. 13 b) Second concurrent data stream tree

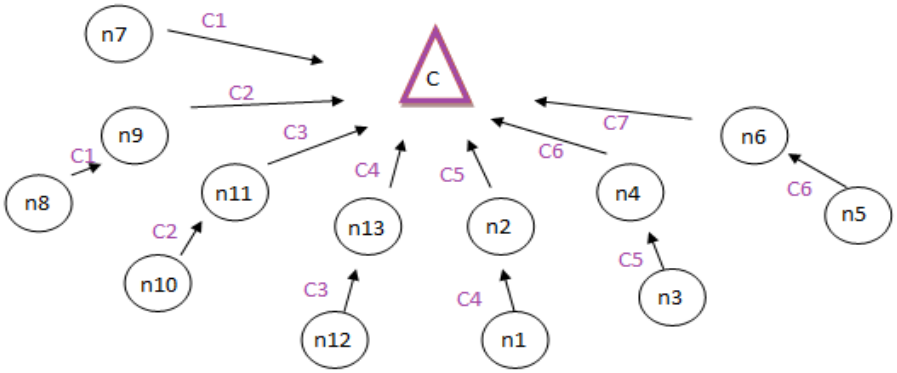

Fig. 13 c) Third concurrent data stream tree

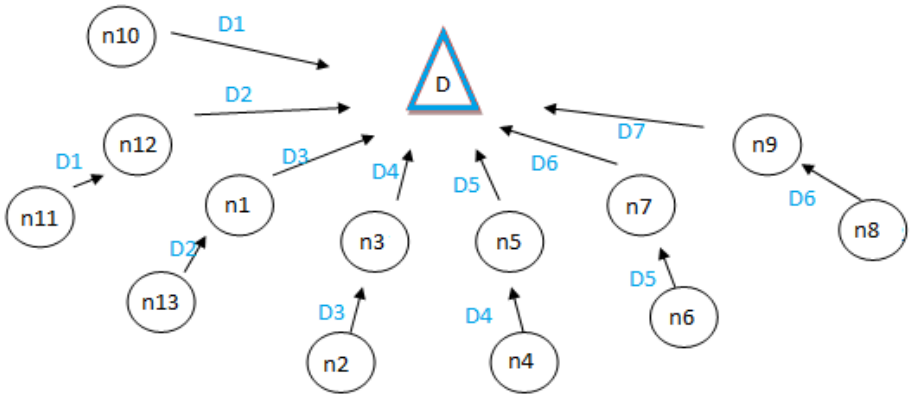

Fig. 13 d) Fourth concurrent data stream tree

Fig. 14 Tree structures of the α-ring (2nd &3rd ring)

Fig. 14 a) First concurrent data stream tree

Fig. 14 b) Second concurrent data stream tree

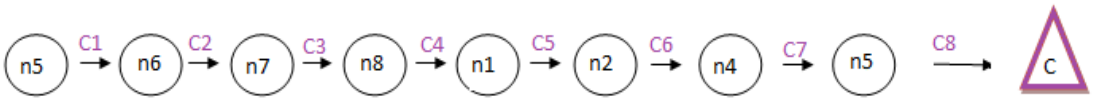

Fig. 14 c) Third concurrent data stream tree

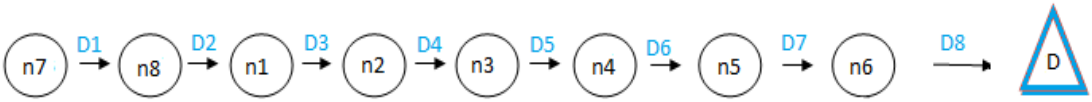

Fig. 14 d) Fourth concurrent data stream tree

- Structure of both the α-rings will be same as above as they have same number of nodes.

4.2.1.2. b) Node fault

Cases:

1. **If** fault occurs in one of the node of β-ring. For example node 11 of β-ring gets faulty:
    - The faulty node will be deleted.
    - We will compute whether β-ring still have minimum of 3k nodes because this the necessary conditions for β-ring to be formed.
        i. In this case the condition is met.

ii. Even after deletion of the faulty nodes, the number of nodes left are 12 which is equal to 3k (where k=4).

iii. Therefore, we just have to re-arrange the structure without any computations.

- The tree structure after deletion of the nodes will be as follow

Fig. 15 Tree structures of β-ring after deletion of faulty node

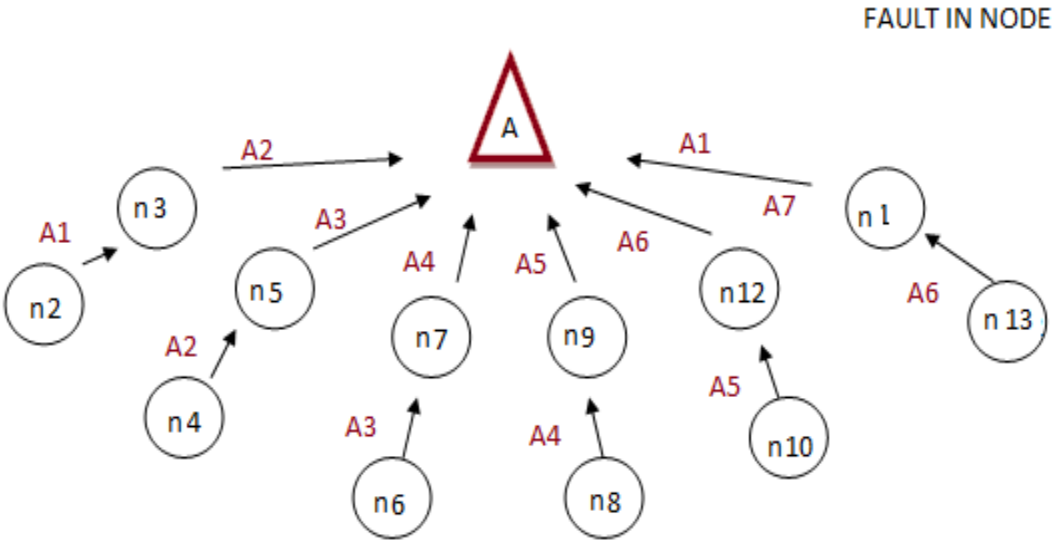

Fig. 15 a) First concurrent data stream tree

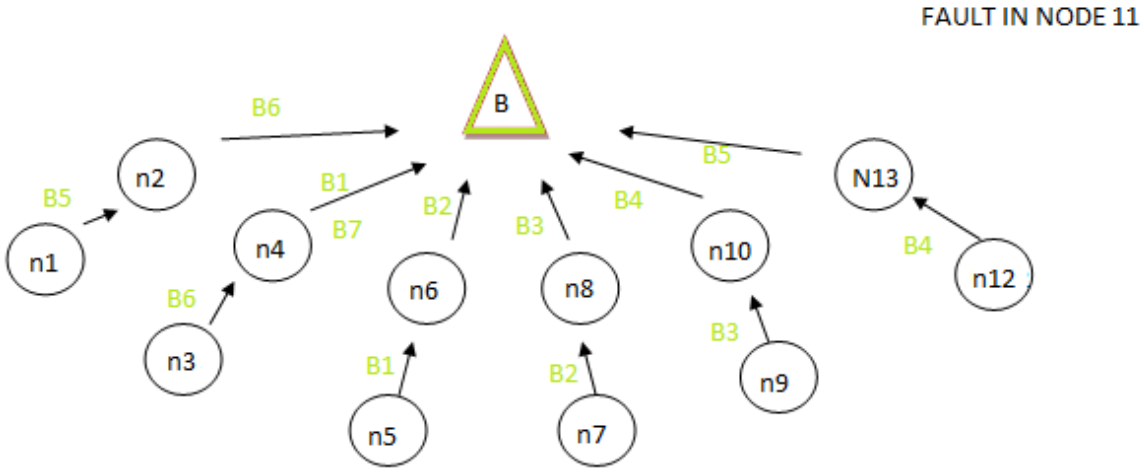

Fig. 15 b) Second concurrent data stream tree

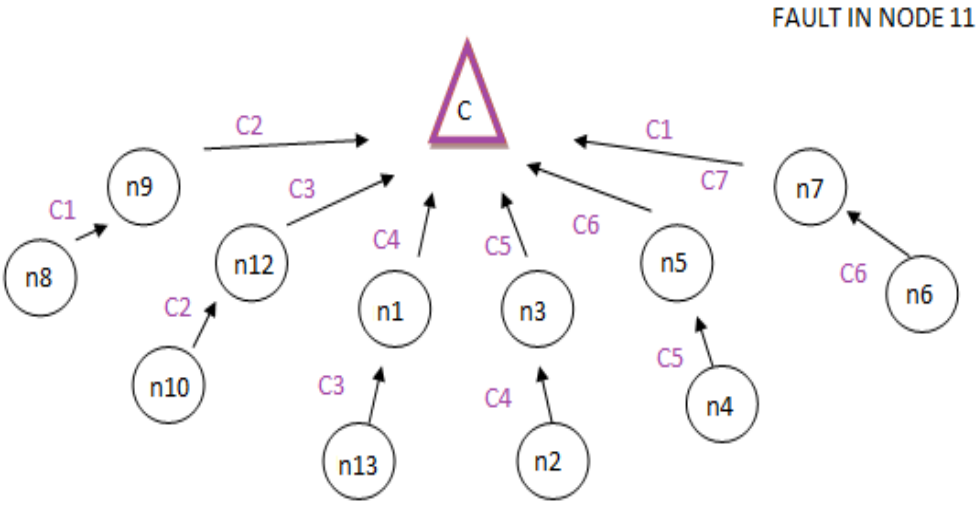

Fig. 15 c) Third concurrent data stream tree

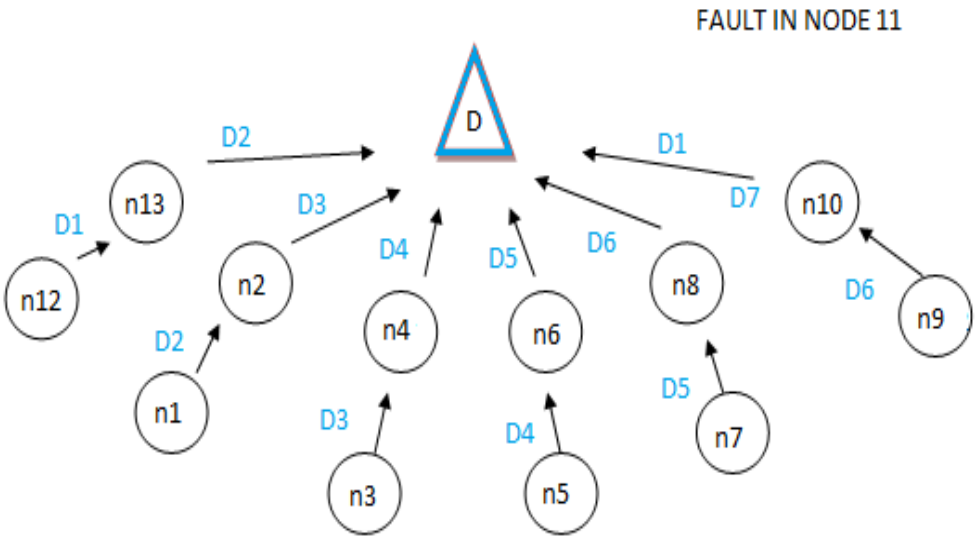

Fig. 15 d) Fourth concurrent data stream tree

- There will be no change in rest of the two rings
- The structures of both the α-rings will be same.

2. Else if node fault occurs in one of the α-ring. For example node 4 of 2nd ring(α-ring) gets faulty
    - Since both the α-rings have same number of nodes
        i. The last node of the β-ring will be inserted at the end of the 2nd ring and will be deleted from its original ring.
    - The tree structures of the all the rings after deletion of the faulty node will be as follows:

Fig. 16 Tree structures of the β-ring after deletion of faulty ring

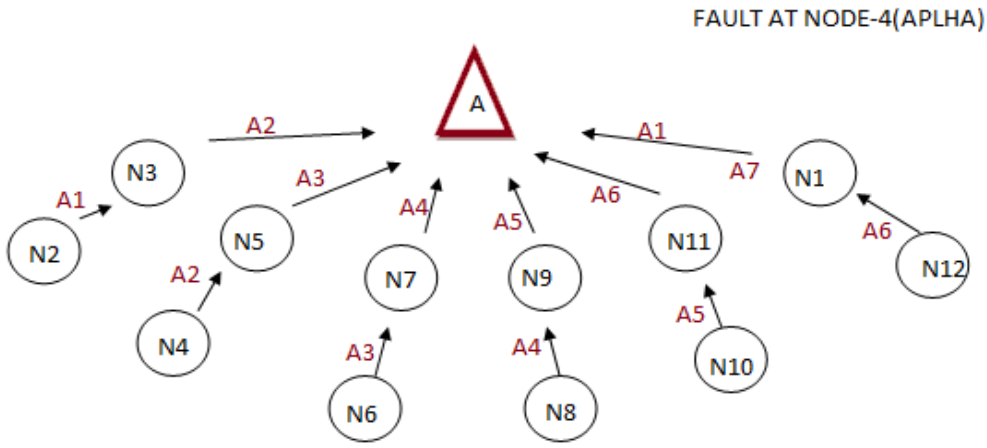

Fig. 16 a) First concurrent data stream tree

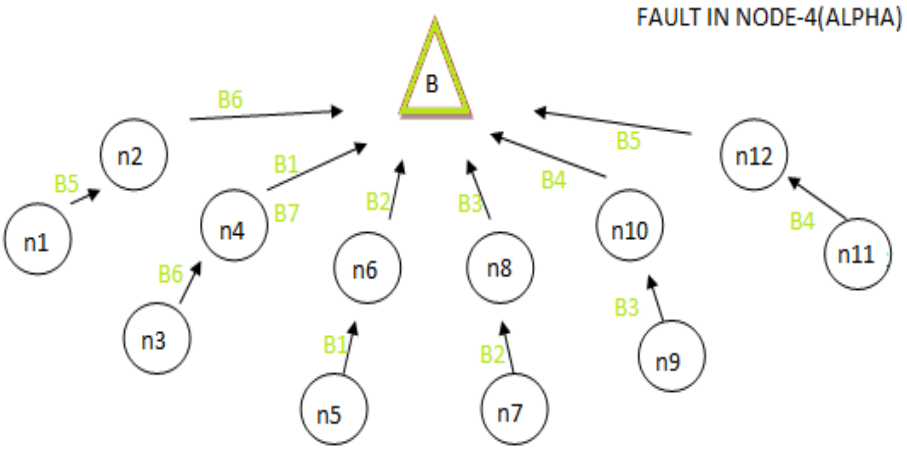

Fig. 16 b) Second concurrent data stream tree

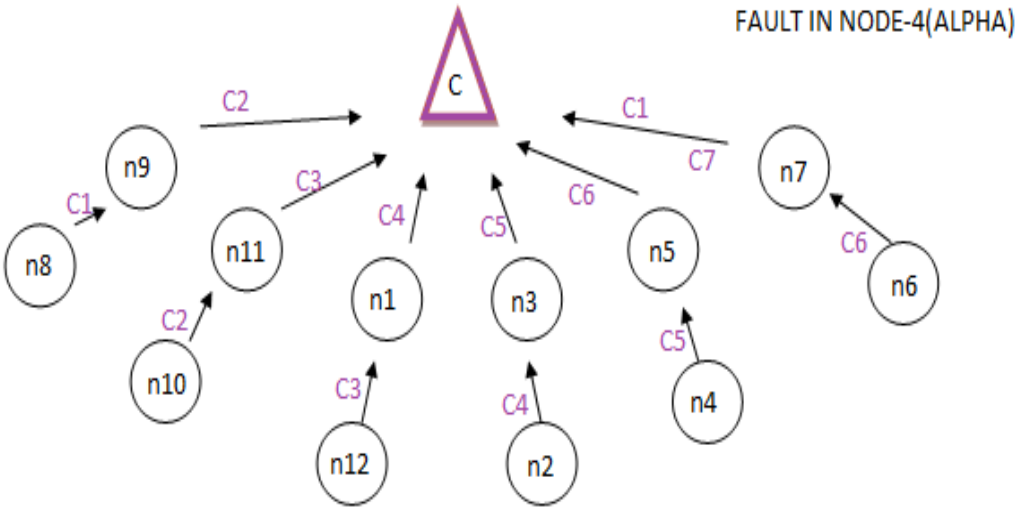

Fig. 16 c) Third concurrent data stream tree

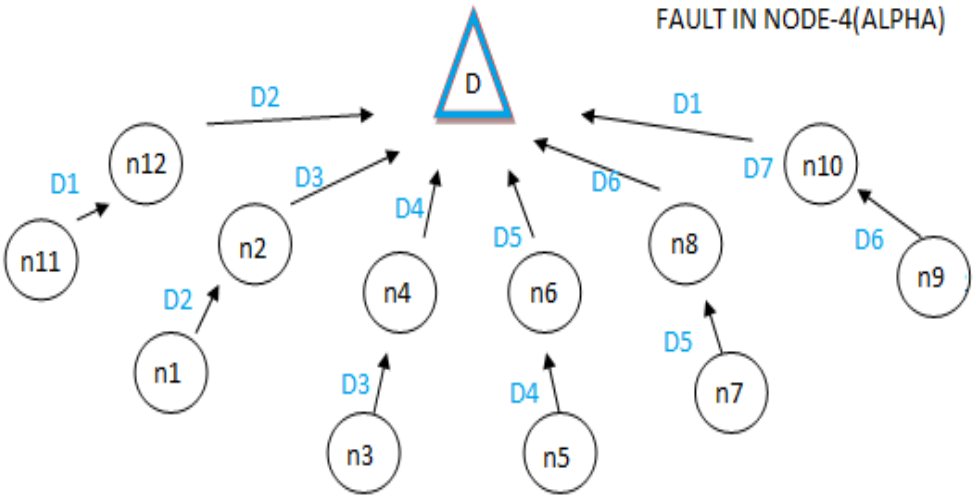

Fig. 16 d) Fourth concurrent data stream tree

Fig. 17 Tree structures of the α-ring after deletion of faulty ring

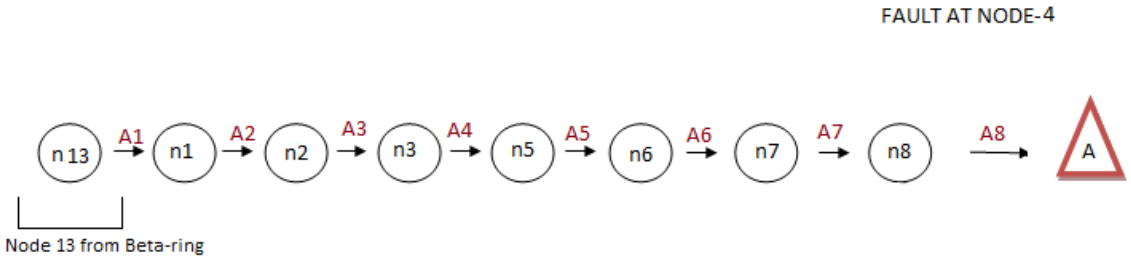

Fig. 17 a) First concurrent data stream tree

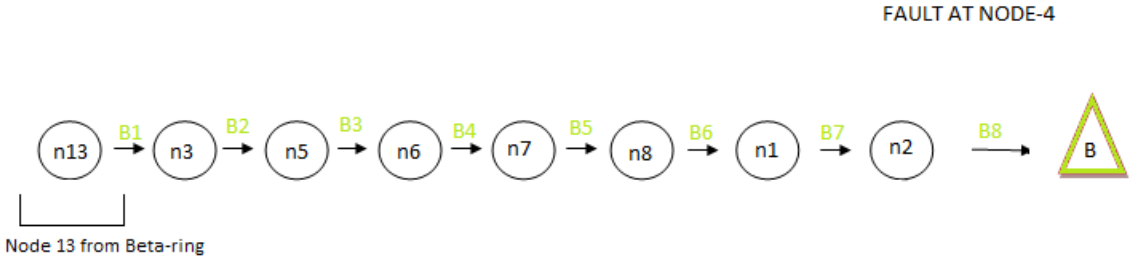

Fig. 17 b) Second concurrent data stream tree

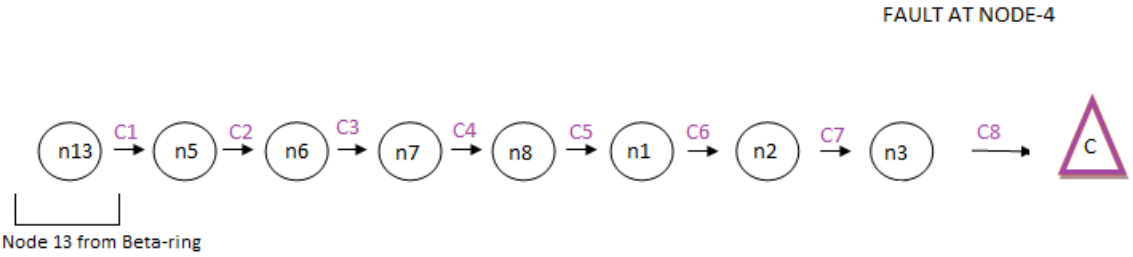

Fig. 17 c) Third concurrent data stream tree

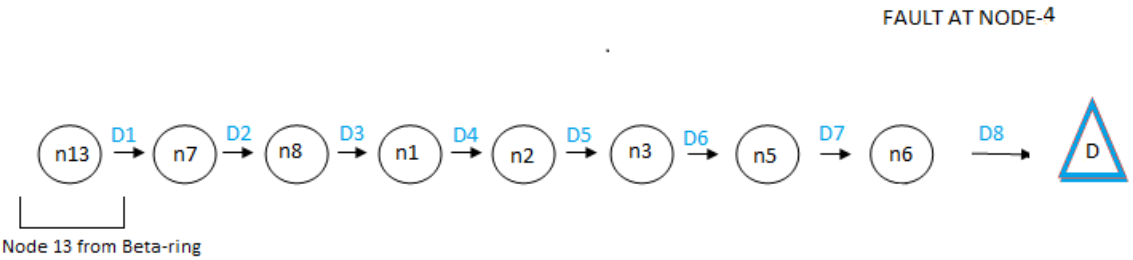

Fig. 17 d) Fourth concurrent data stream tree

- There will be no change in 3rd ring (α-ring)
- Therefore the structure of the 3rd ring remains unchanged.

4.2.1.3 Example 3

|N|=30 k=4 $u_{max}= \lfloor 30/4 \rfloor=7$

4.2.1.3 a) Structure formation

- u_{max} is odd (multiple rings (both α and β rings would be formed).
- T1=6, T2=3; T1+T2=9.
- 1β-ring with be formed.
- The β-ring with 3k nodes will be formed i.e. 12 nodes
- (umax-3)/2 α- rings i.e. 2 α-rings will be formed.
- Each α-ring will contain minimum of 2k nodes i.e. 8 nodes

  12 8 8 (12+8+8=28)

- Two nodes are still left so it will be assigned to the β-ring until $N_\beta=2T_1+1$.

  12 +1 8 8

  13 8 8 (13+8+8=29)

- After assigning one more node to β-ring, $N_\beta=13$. Therefore we cannot assign the remaining node to the β-ring.
- We need to assign the last node to first of the α-ring.

  13 8 +1 8

  13 9 8

- β-ring contains 13 nodes
- 1st α-rings contain 9 nodes (alpha_max).
- 2nd α-rings contain 8 nodes (alpha_min).
- The ring structures will be as follows:

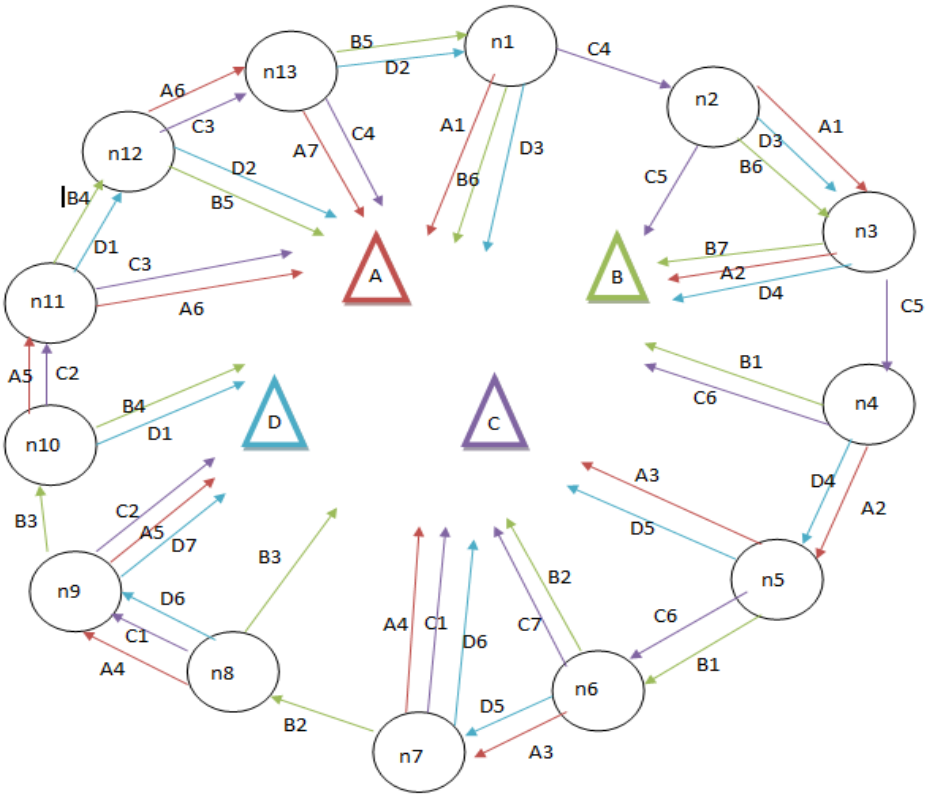

Fig. 18 β-ring structure with 13 nodes

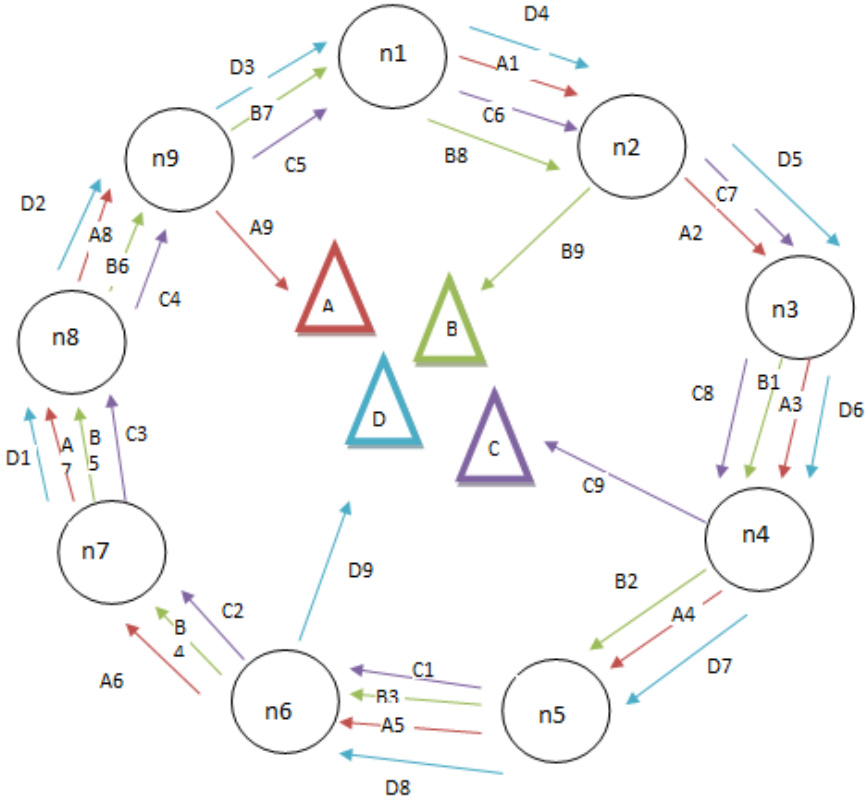

Fig. 19 α-ring structure with 9 nodes

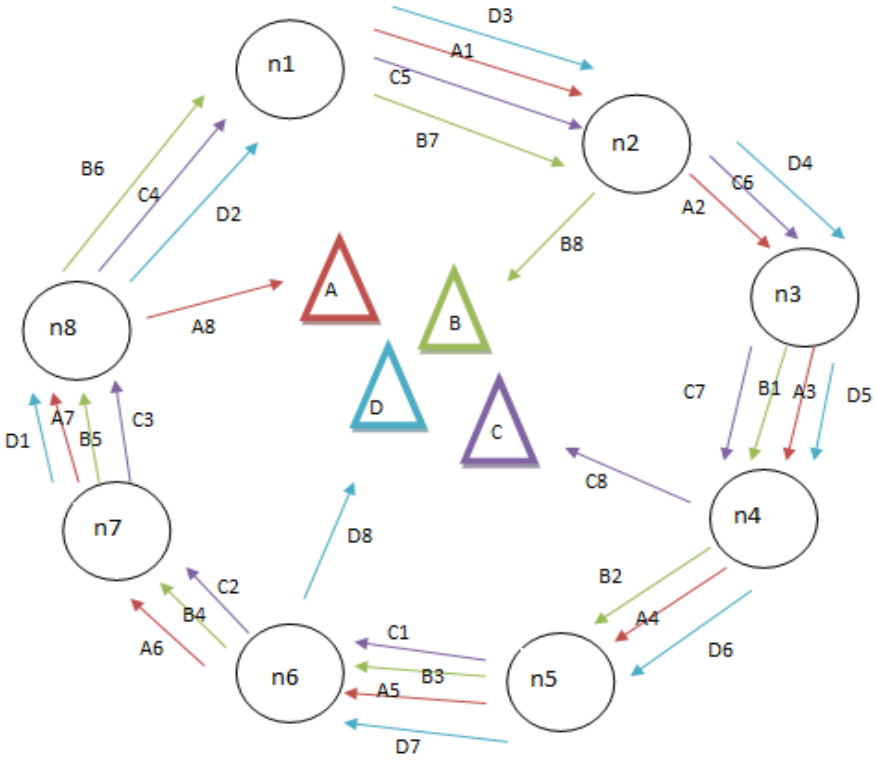

Fig. 20 **α-ring structure with 8 nodes**

Fig. 21 **Tree structures of β-ring**

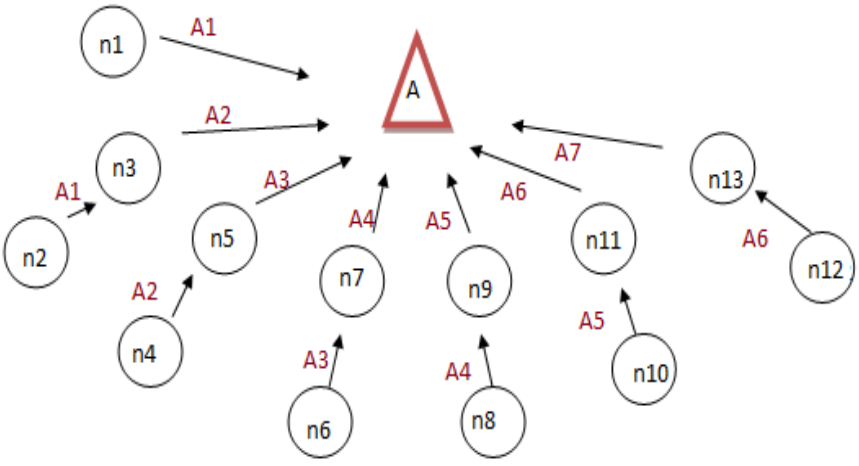

Fig. 21 a) First concurrent data stream tree

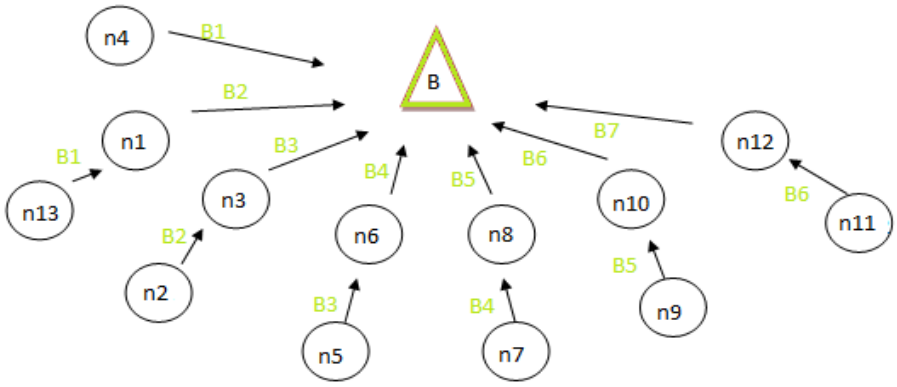

Fig. 21 b) Second concurrent data stream tree

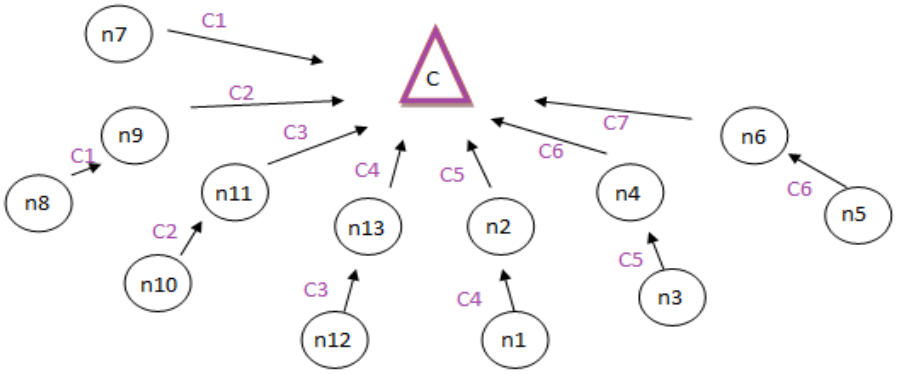

Fig. 21 c) Third concurrent data stream tree

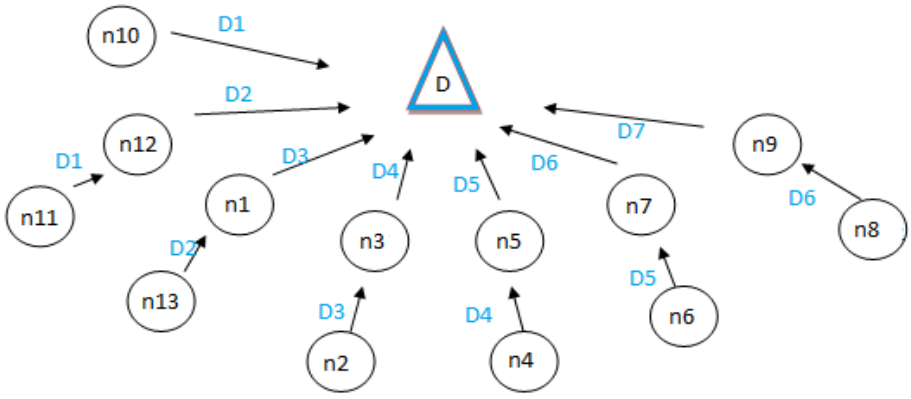

Fig. 21 d) Fourth concurrent data stream tree

Fig. 22 Tree structures of 1st α-ring (alpha_max)

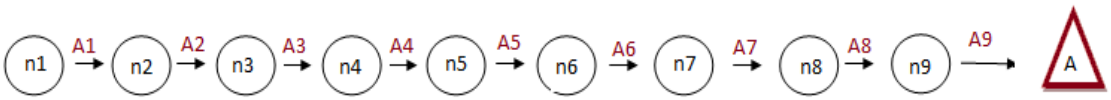

Fig. 22 a) First concurrent data stream tree

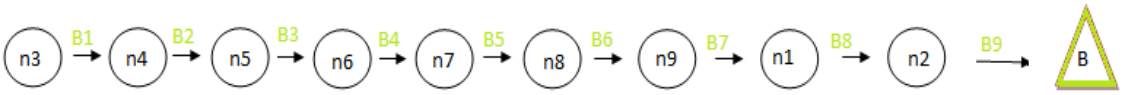

Fig. 22 b) Second concurrent data stream tree

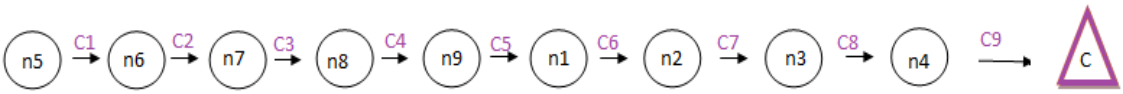

Fig. 22 c) Third concurrent data stream tree

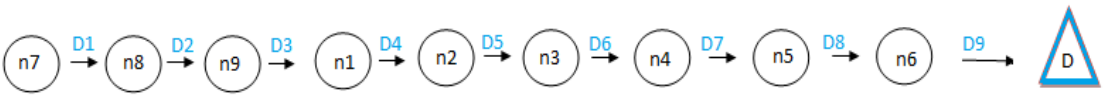

Fig. 22 d) Fourth concurrent data stream tree

Fig. 23 Tree structures of 2nd α-ring (alpha_min)

Fig. 23 a) First concurrent data stream tree

Fig. 23 b) Second concurrent data stream tree

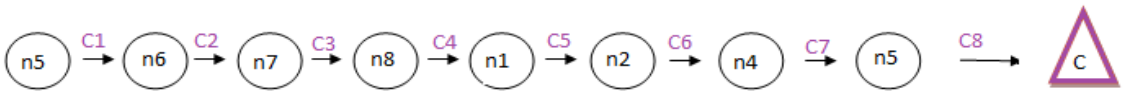

Fig. 23 c) Third concurrent data stream tree

Fig. 23 d) Fourth concurrent data stream tree

4.2.1.3. b) Node fault

Cases:

1. **If** fault occurs in one of the node of β-ring. For example node 11 of β-ring gets faulty:
    - This would be same as Example 2: Case1
    - The tree structure after deletion of the nodes will be same as figure
2. **Else if** fault occurs in one of the node of alpha_max i.e. 1st α-ring. For example node 4 gets faulty:
    - Simply the faulty node is deleted.
    - The α-ring is re-arranged with one node less.
    - The tree structures of all the rings after deletion of faulty node is as follows

Fig. 24 After faulty node deletion tree structures of the apha_max ring

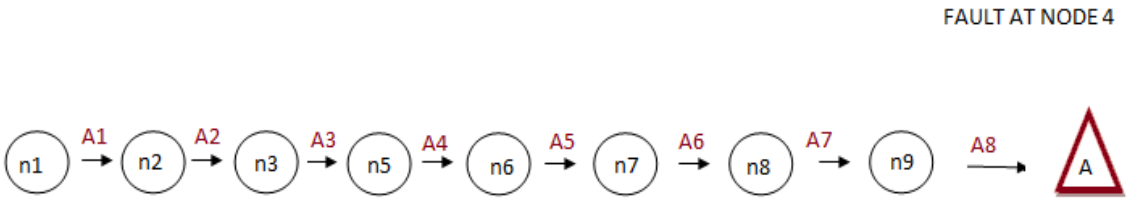

Fig. 24 a) First concurrent data stream tree

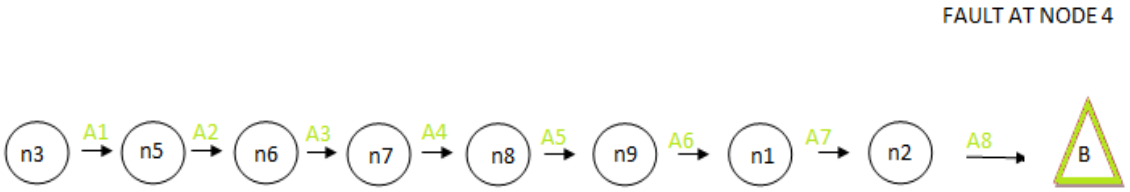

Fig.24 b) Second concurrent data stream tree

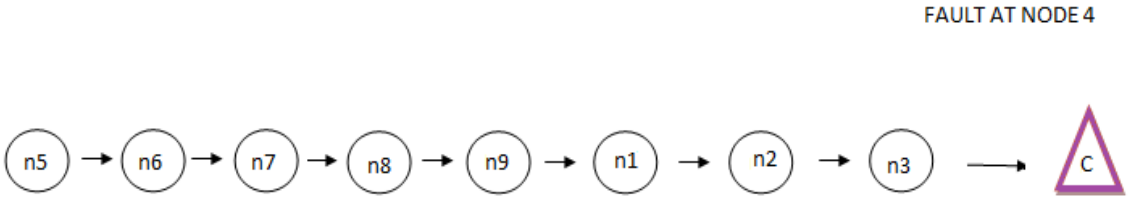

Fig. 24 c) Third concurrent data stream tree

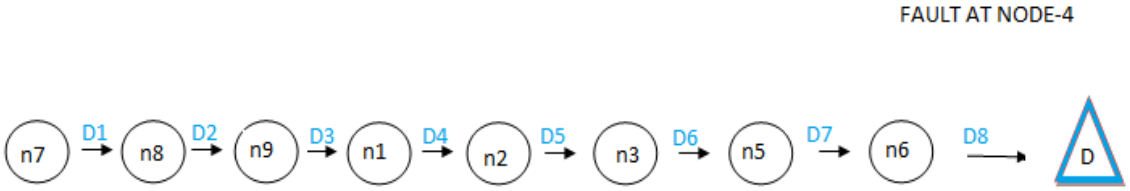

Fig. 24 d) Fourth concurrent data stream tree

- The structure of rest of the two rings will remain same.

3. **Else if** fault occurs in one of the node of alpha_min i.e. 2nd α-ring. For example node 6 gets faulty:
    - We will delete the faulty node.
    - If after deletion the number of nodes in alpha_min is less than 2k:
        i. In this case after deletion the number of nodes in alpha_min is equal to 7 which is less than 2k nodes (8 nodes).
        ii. Therefore, the last node of alpha_max has to be inserted at the end of alpha_min and has to deleted from the orifinal tree to which it belongs.
        iii. Now the apha_max will have equal number of nodes to alpha_min i.e. 8 in this case.
    - The tree structures of all the rings after deletion of faulty node is as follows:

Fig. 25 After faulty node deletion tree structures of the apha_min ring

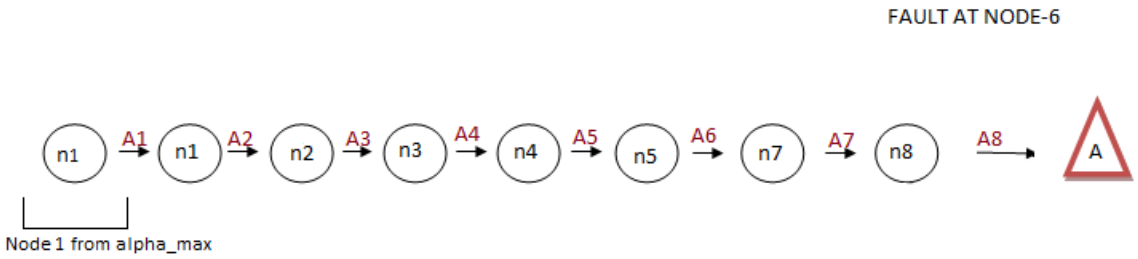

Fig. 25 a) First concurrent data stream tree

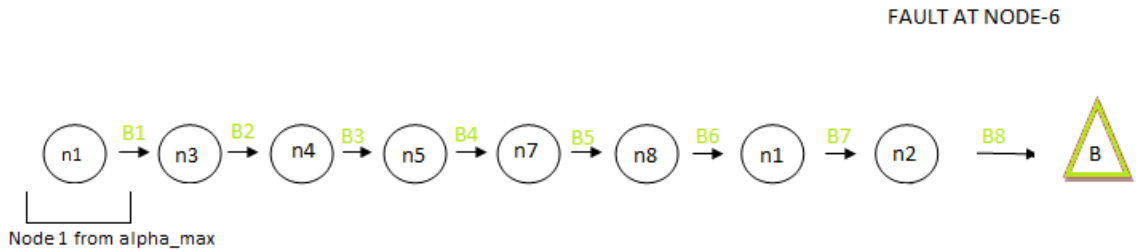

Fig. 25 b) Second concurrent data stream tree

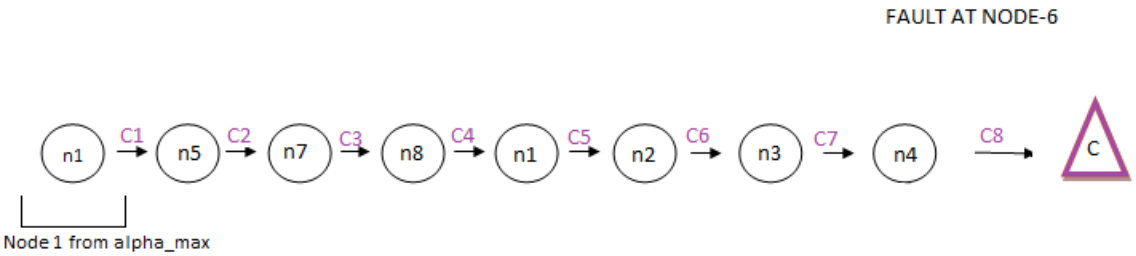

Fig. 25 c) Third concurrent data stream tree

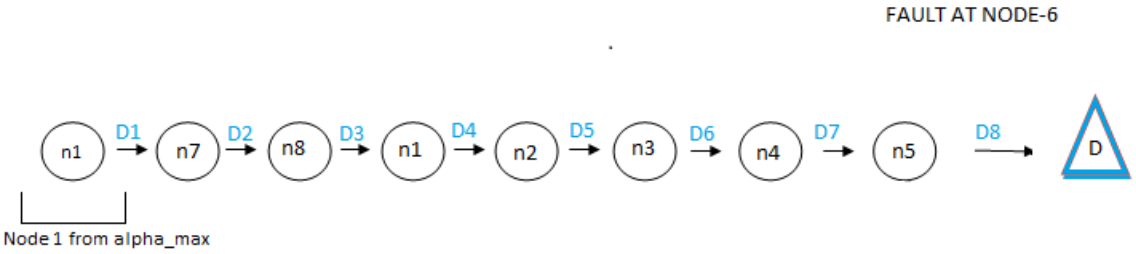

Fig. 25 d) Fourth concurrent data stream tree

Fig. 26 After faulty node deletion tree structures of the alpha_max ring

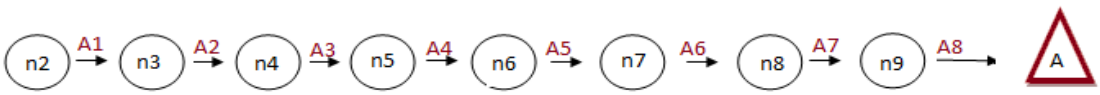

Fig. 26 a) First concurrent data stream tree

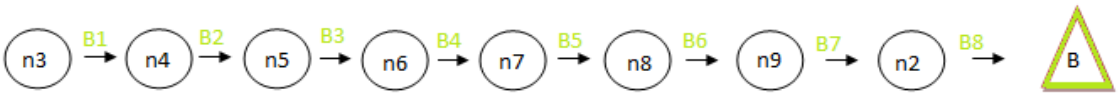

Fig. 26 b) Second concurrent data stream tree

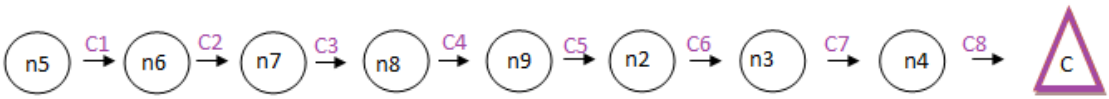

Fig. 26 c) Third concurrent data stream tree

Fig. 26 d) Fourth concurrent data stream tree

- The structure of the β-ring would be same, without any change.

4.2.1.4 Example 4

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

4.2.1.4. a) Structure formation

- u_{max} is odd (multiple rings (both α and β rings would be formed).
- T1=4, T2=4; T1+T2=8.
- 1 β-ring with 3k nodes will be formed i.e. 9 nodes
- (umax-3)/2 α- rings i.e. 3 α-rings will be formed.
- Each α-ring will contain minimum of 2k nodes i.e. 6 nodes

9 6 6 6 (9+6+6+6=27)

- One node is still left so it will be assigned to the β-ring until $N_\beta=2T_1+1$.
- But in this case $N_\beta=9$ which is == $2T_1+1$.
- Therefore, the remaining node will be assigned to the 1st α-ring.

9 6+1 6 6

9 7 6 6

- β-ring contains 9 nodes
- 1st α-ring (alpha_max) contains 7 nodes.
- Other two α-rings (apha_min) contain 6 nodes each.
- The ring structures will be as follows:

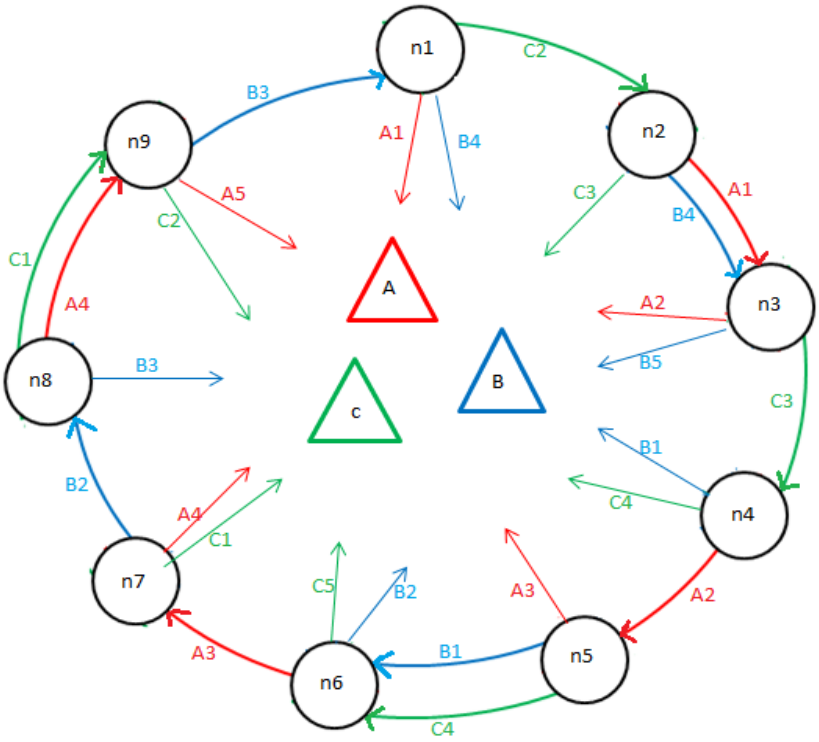

Fig. 27 β-ring with 9 nodes

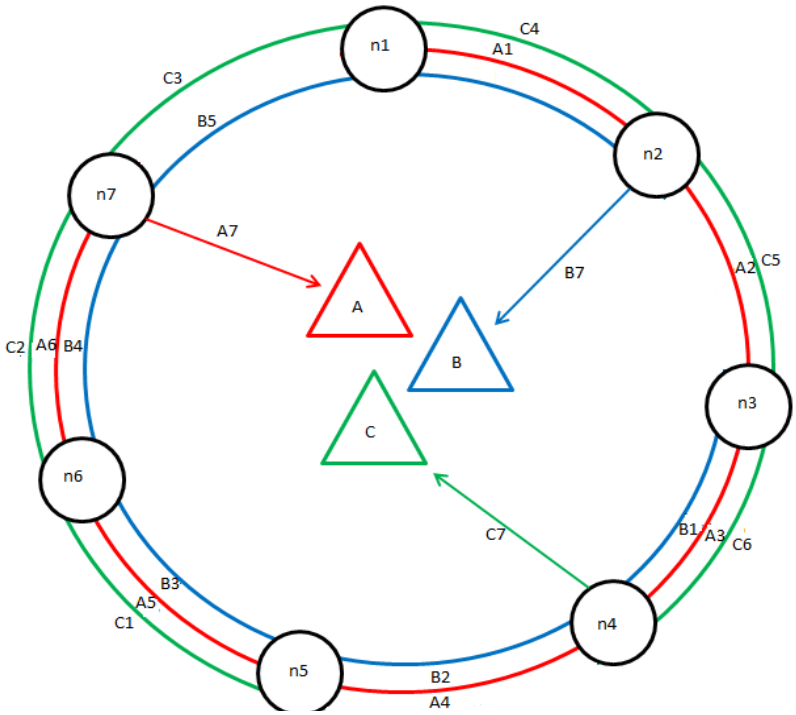

Fig. 28 1st α-ring (alpha_max) with 7 nodes

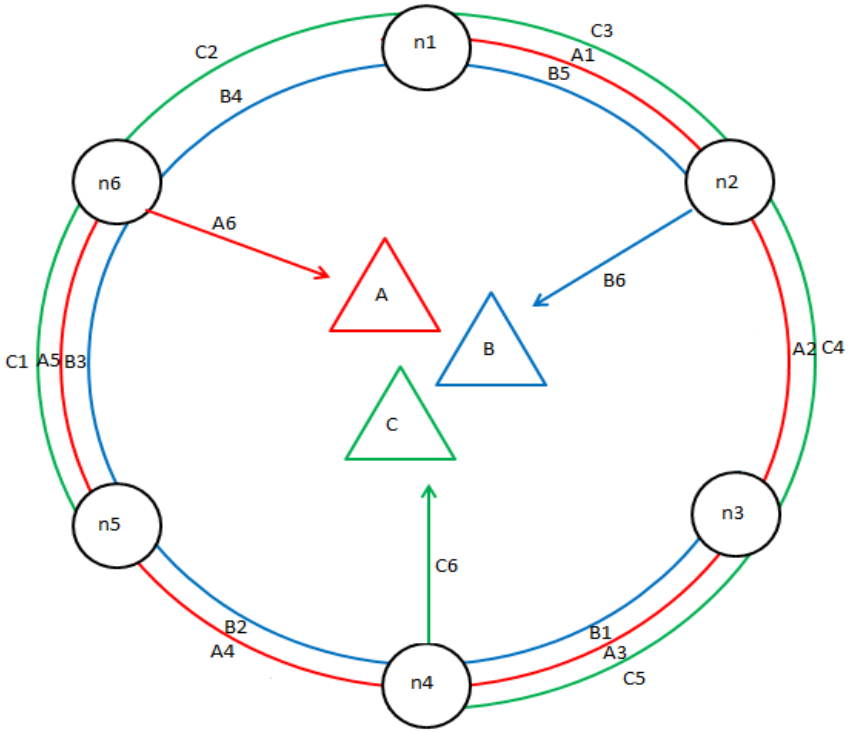

Fig. 29 2nd α-ring (alpha_min) with 6 nodes

- Structure of 3rd and 4th α-rings would be same as 2nd α-ring since the number of nodes are same and equal to 6.
- Tree structures of all the rings are as follows:

Fig. 30 Tree structures of the β-ring

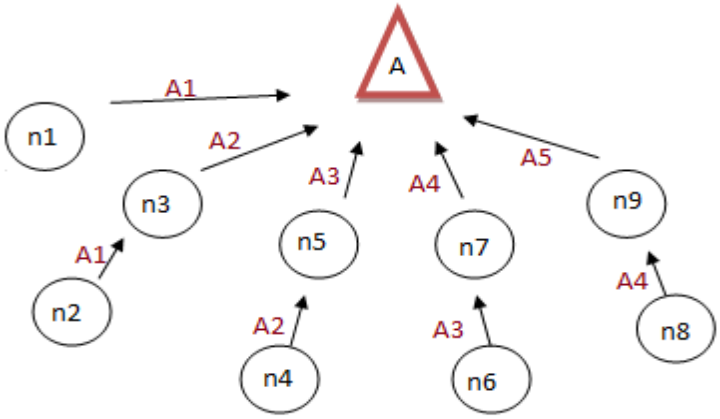

Fig. 30 a) First concurrent data stream tree

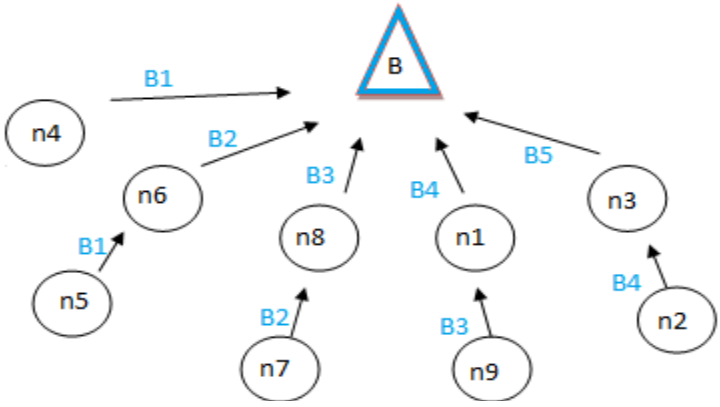

Fig. 30 b) Second concurrent data stream tree

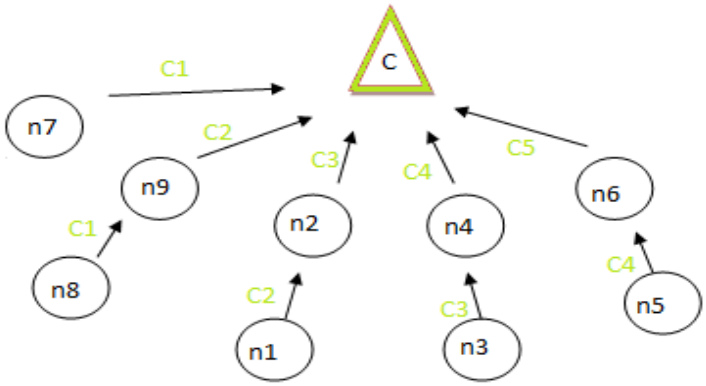

Fig. 30 c) Third concurrent data stream tree

Fig. 31 Tree structures of 1st α-ring (alpha_max)

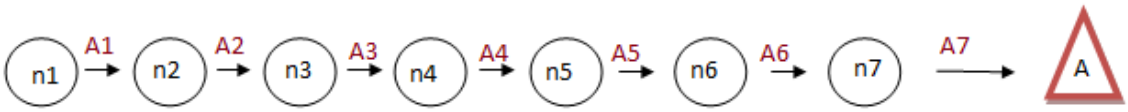

Fig. 31 a) First concurrent data stream tree

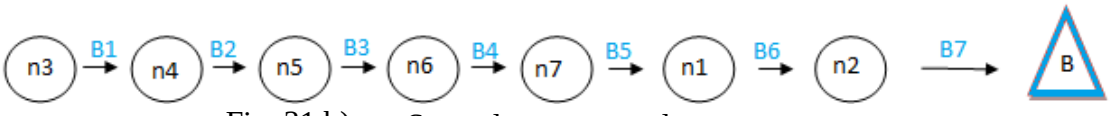

Fig. 31 b) Second concurrent data stream tree

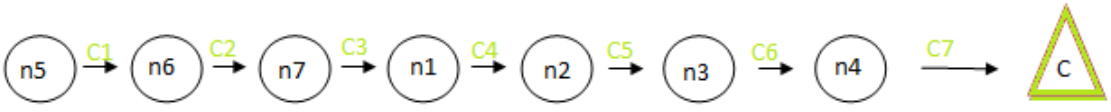

Fig. 31 c) Third concurrent data stream tree

Fig. 32 Tree structures of 2nd α-ring (alpha_min)

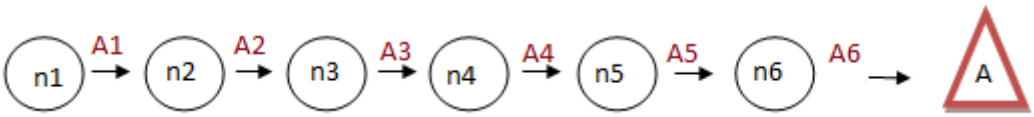

Fig. 32 a) First concurrent data stream tree

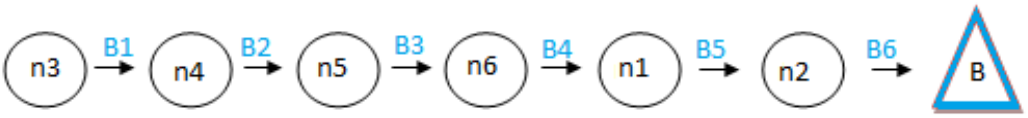

Fig. 32 b) Second concurrent data stream tree

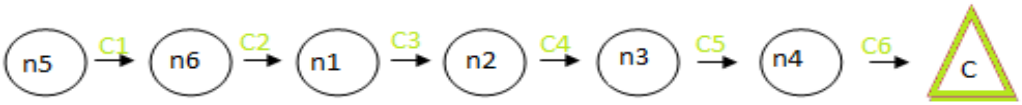

Fig. 32 c) Third concurrent data stream tree

- Tree structures of the remaining 2 α-rings would be same as 2nd α-ring (alpha_min) due to the same number of nodes.

4.2.1.4. b) Node faulty

Cases:

1. **If** fault occurs in one of the node of β-ring. For example node 8 of β-ring gets faulty:
    - The faulty node will be deleted.
    - We will compute whether β-ring still have minimum of 3k nodes because this the necessary conditions for β-ring to be formed.
        i. In this case the condition is **not** met.
        ii. After deletion of the faulty nodes, the number of nodes left are 8 which is less than 3k (where k=3).
        iii. Therefore, the last node of the alpha_max ring will be inserted at the end of β-ring and will be deleted from its original ring.

- The tree structure after deletion of the nodes will be as follow

Fig. 33 Tree structures of β-ring after deletion of faulty node

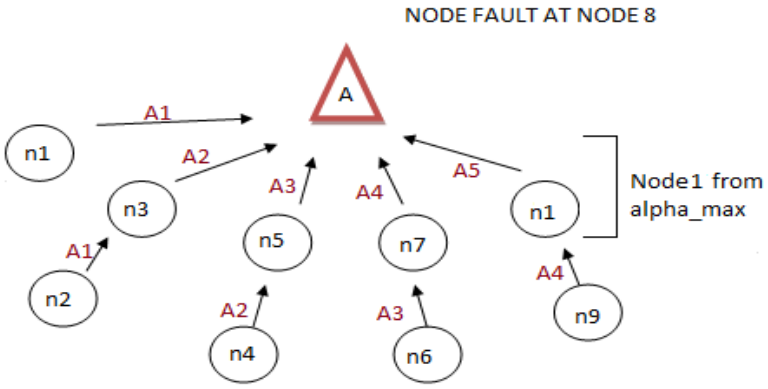

Fig. 33 a) First concurrent data stream tree

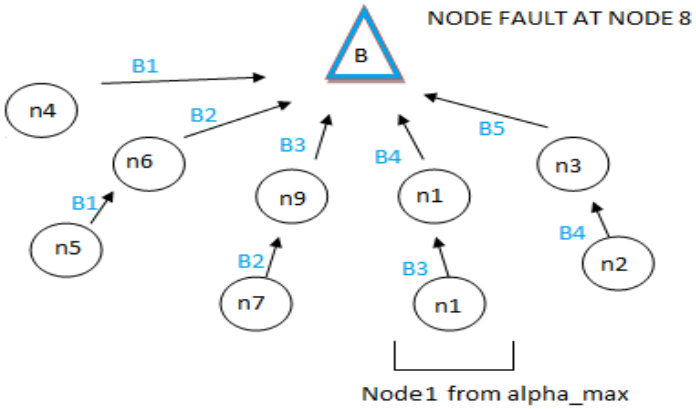

Fig. 33 b) Second concurrent data stream tree

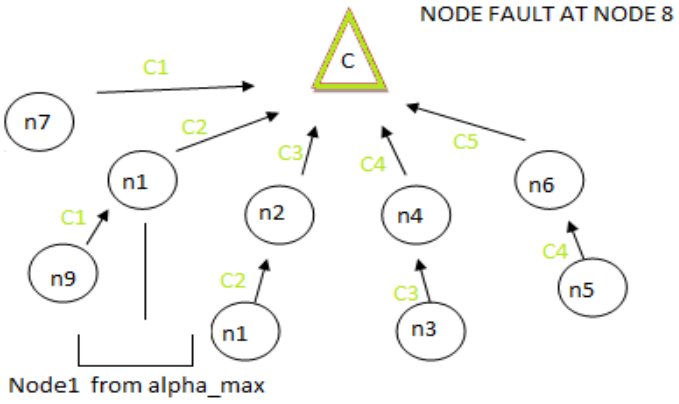

Fig. 33 c) Third concurrent data stream tree

Fig. 34 After deletion of faulty node 1st α-ring's tree structures

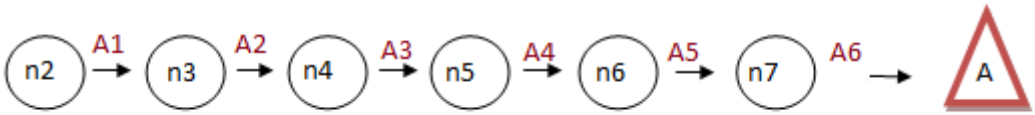

Fig. 34 a) First concurrent data stream tree

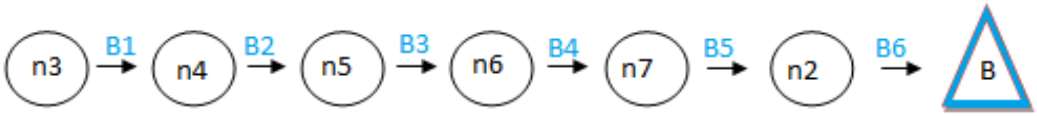

Fig. 34 b) Second concurrent data stream tree

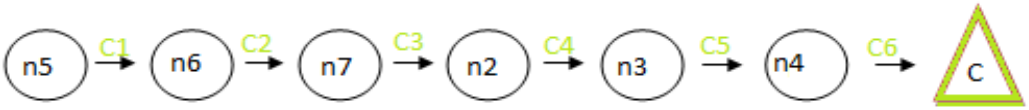

Fig. 34 c) Third concurrent data stream tree

- The structures of rest of the rings will remain as before.

2. **Else if** fault occurs in one of the node of 1st α-ring i.e. alpha_max.
    - This would be same as Example 3: Case 2
    - Simply the faulty node is deleted.
    - The α-ring is re-arranged with one node less.
    - The working and re –arrangement would be similar to Example 3: Case 2
3. **Else if** node fault occurs in one of the node of 2nd or 3rd α-rings i.e. alpha_min.
    - This would be same as Example 3: Case 2
    - We will delete the faulty node.
    - If after deletion the number of nodes in alpha_min is less than 2k, like in this case
        i. The working and re –arrangement would be similar to Example 3: Case 3

4.2.2 Cases where u_{max} changes

When u_{max} changes after deletion of the faulty node and therefore, there is a transition from α-ring structure to β-ring/Multiple ring structure and vice-versa.

4.2.2.1 Example 5 (Multiple ring to α-ring)

|N|=27 k=3 u_{max}= ⌊ 27/3 ⌋=9

4.2.2.1. a) Structure Formation

- u_{max} is odd (multiple rings (both α andβ rings would be formed).
- T1=4, T2=3; T1+T2=7.
- 1β-ring with 3k nodes will be formed i.e. 9 nodes
- (u_{max}-3)/2 α- rings i.e. 3 α-rings will be formed.
- Each α-ring will contain minimum of 2k nodes i.e. 6 nodes

  9 6 6 6 (9+6+6+6=27)
- β-ring contains 9 nodes
- All the α-rings contain 6 nodes each.
- The ring structures will be as follows:

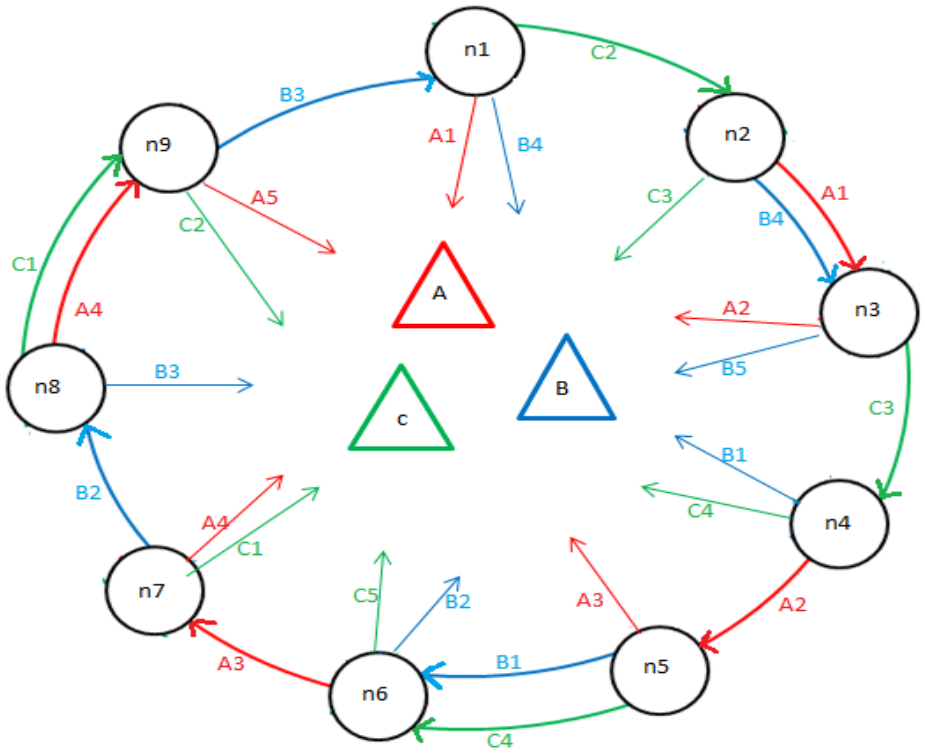

Fig. 35 β-ring with 9 nodes

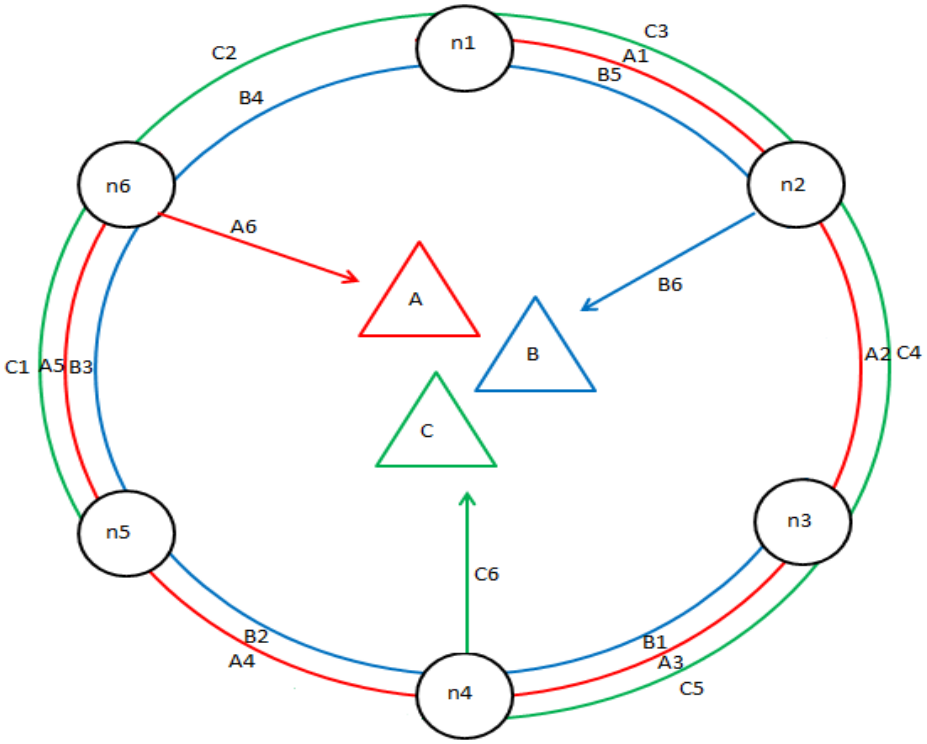

Fig. 36 1st α-ring (alpha_min) with 6 nodes

- Structure of rest of the α-rings would be same as 1st α-ring since the number of nodes is same that is equal to 6.
- Tree structures of all the rings are as follows:

Fig. 37 Tree structures of the β-ring

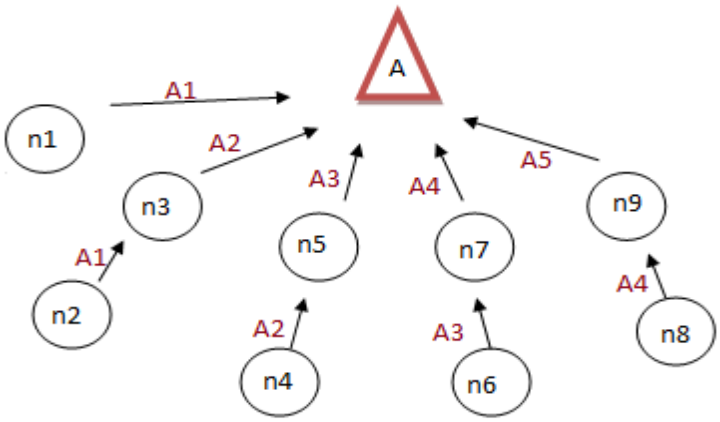

Fig. 37 a) First concurrent data stream tree

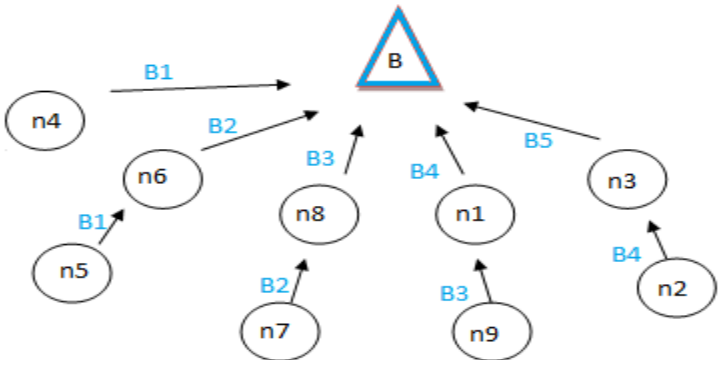

Fig. 37 b) Second concurrent data stream tree

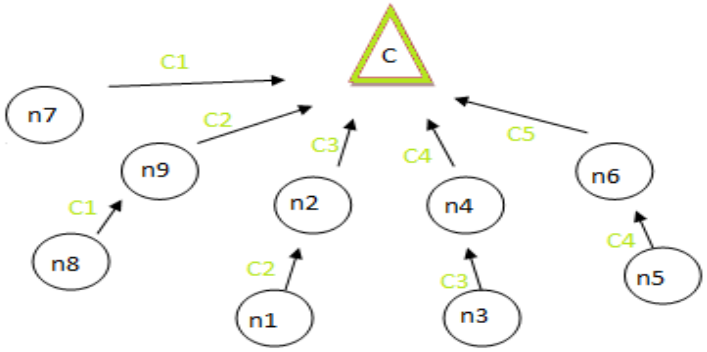

Fig. 37 c) Third concurrent data stream tree

Fig. 38 Tree structures of 2nd α-ring (alpha_min)

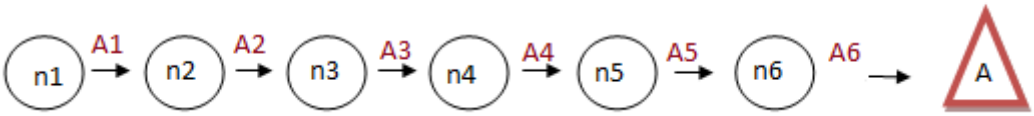

Fig. 38 a) First concurrent data stream tree

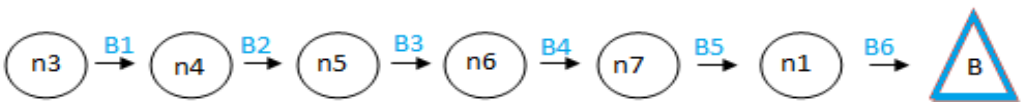

Fig. 38 b) Second concurrent data stream tree

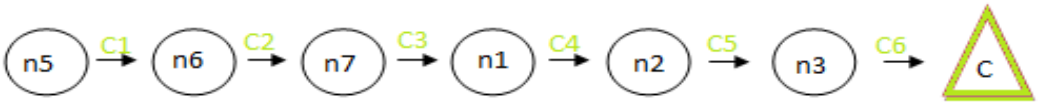

Fig. 38 c) Third concurrent data stream tree

- Tree structures of the remaining α-rings would be same as 1st α-ring due to the same number of nodes.

4.2.2.1. b) Node faulty

- **If** single node becomes faulty in any of the ring:
  i. u_{max} changes from odd to even.
  ii. When u_{max} is even, only α-rings would be formed
  iii. Therefore, there is a transition from multiple rings to only α-rings.

Formation of α- rings

|N|=26 k=3 $u_{max} = \lfloor 26/3 \rfloor = 8$

- T1=5, T2=3; T1+T2=8.
- u_{max} /2 α- rings i.e. 4 α-rings will be formed.
- Each α-ring will contain minimum of 2k nodes i.e. 6 nodes

6 6 6 6 (6+6+6+6=24)

- The remaining 2 nodes will be assigned to the α-rings one by one.

6 +1 6+1 6 6 (7+7+6+6=26)

- The 1st and 2nd α-rings contain 7 nodes each (alpha_max).
- The 3rd and 4th α-rings contain 6 nodes each (alpha_min).
- The ring structures will be as follows:

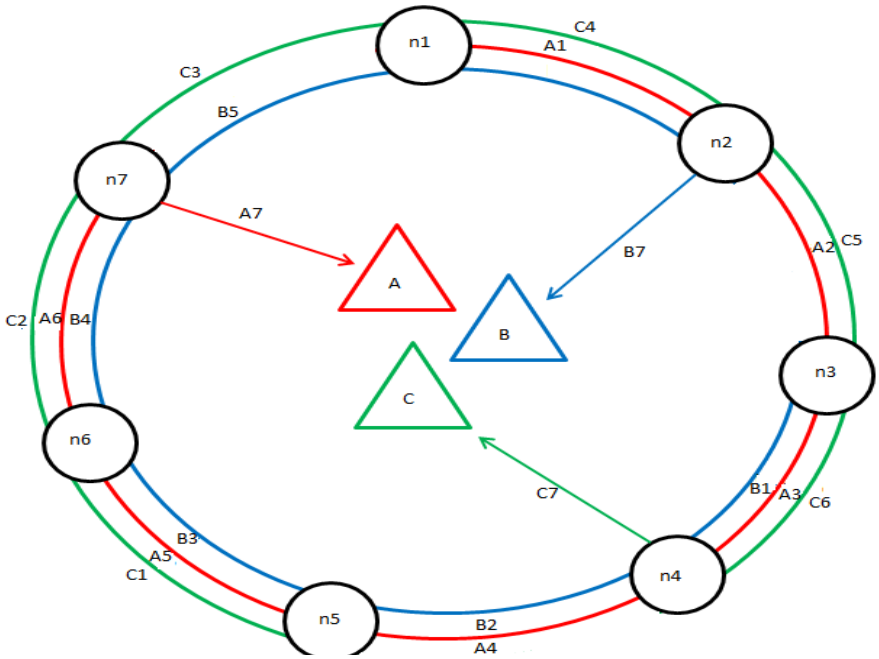

Fig. 39 1st α-ring (alpha_max) with 7 nodes

- The ring structure of 2nd α-ring (alpha_max) would be same as 1st α-ring because they have same number of nodes.

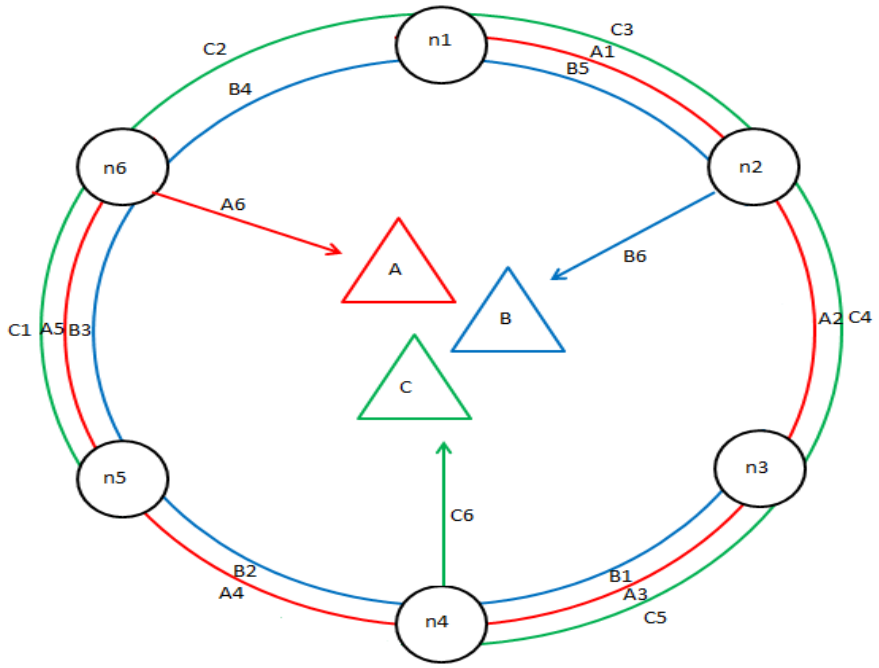

Fig. 40 3rd α-ring (alpha_min) with 6 nodes

- The ring structure of 4th α-ring (alpha_min) would be as same as 3rd α-ring because they have same number of node.
- The tree structures of the rings are as follows:

Fig. 41 Tree structures of 1st α-ring (alpha_max)

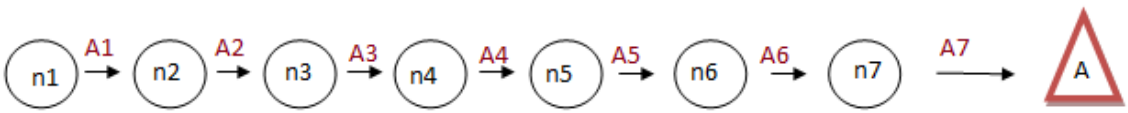

Fig. 41 a) First concurrent data stream tree

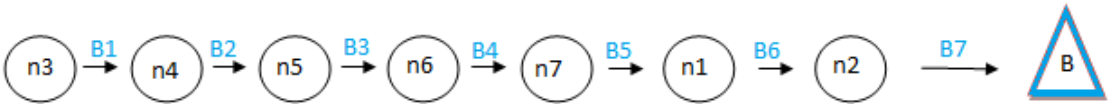

Fig. 41 b) Second concurrent data stream tree

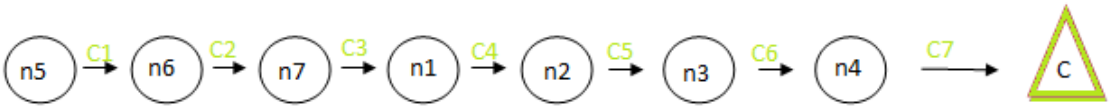

Fig. 41 c) Third concurrent data stream tree

- The tree structure of 2nd α-ring (alpha_max) would be same as 1st α-ring because they have same number of nodes.

Fig. 42 Tree structures of 3rd α-ring (alpha_min)

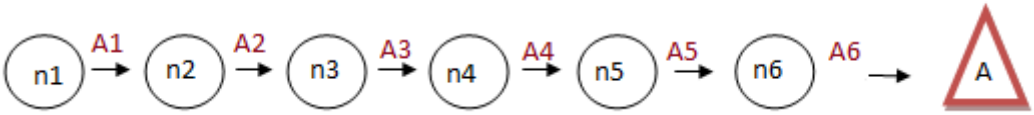

Fig. 42 a) First concurrent data stream tree

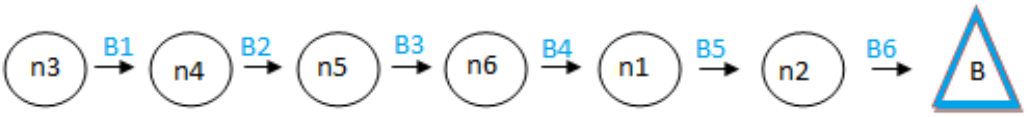

Fig. 42 b) Second concurrent data stream tree

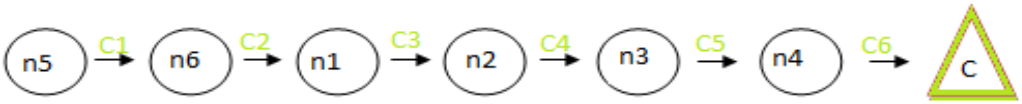

Fig. 42 c) Third concurrent data stream tree

- The ring structure of 4th α-ring (alpha_min) would be as same as 3rd α-ring because they have same number of node.

4.2.2.2 Example 6 (α-ring to Multiple ring)

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

4.2.2.1. a) Structure formation

- u_{max} is even (only α- rings would be formed).
- u_{max} /2 α- rings i.e. 3 α-rings will be formed
- Each α-ring will contain minimum of 2k nodes i.e. 8 nodes

8 8 8 (8+8+8=24)

- All the α-rings will have 8 nodes each.
- T1=7, T2=2; T1+T2=9
- The α-ring structures will be like following

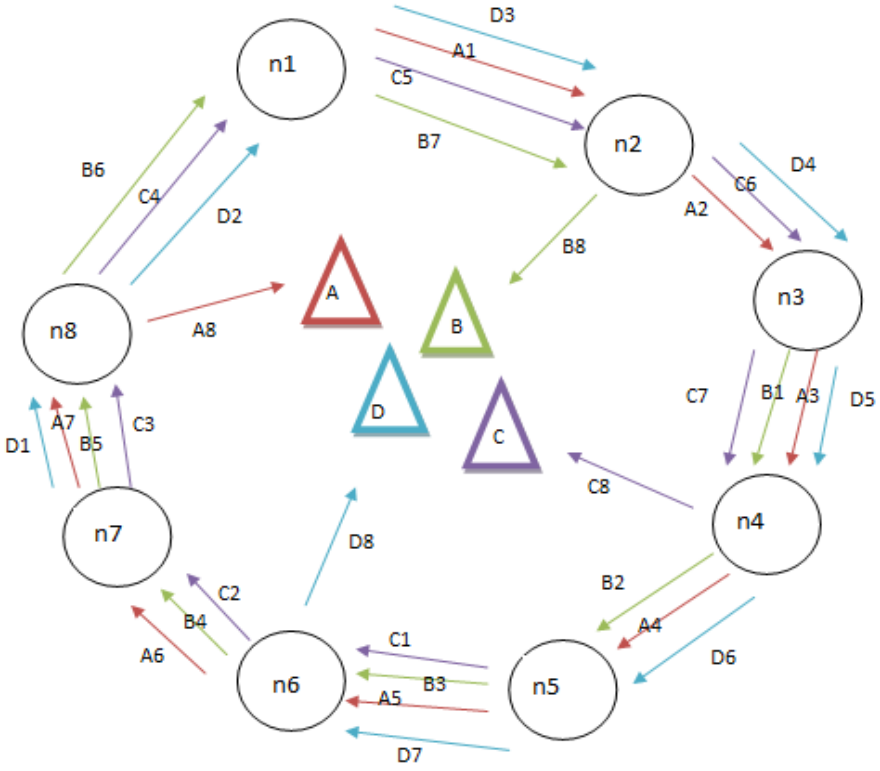

Fig. 43 1st α-ring structure

- The ring structure of rest of the α-rings would be same due to same number of nodes.

- The tree structure of the rings would be as follows:

Fig. 44 Tree structures of 1st α-ring

Fig. 44 a) First concurrent data stream tree

Fig. 44 b) Second concurrent data stream tree

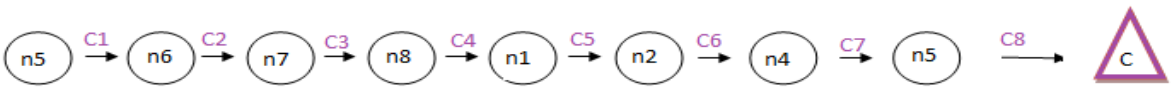

Fig. 44 c) Third concurrent data stream tree

Fig. 44 d) Fourth concurrent data stream tree

- The tree structure of rest of the α-rings would be same due to same number of nodes.

4.2.2.2. b) Node faulty

- **If** single node becomes faulty in any of the ring:
  i. u_{max} changes from even to odd.
  ii. When u_{max} is odd, multiple rings i.e both α and β rings would be formed.
  iii. Therefore, there is a transition from only α-rings to multiple rings.

- **Formation of Multiple rings**

|N|=23 k=4 $u_{max} = \lfloor 23/4 \rfloor = 5$

- u_{max} is odd (multiple rings (both α andβ rings would be formed).
- T1=7, T2=2; T1+T2=9.
- 1 β-ring with 3k nodes will be formed i.e. 12 nodes
- (u_{max} -3)/2 α- rings i.e. 1 α-ring will be formed.
- Each α-ring will contain minimum of 2k nodes i.e. 8 nodes

12 8 (12+8=20)

- Two nodes are still left so it will be assigned to the β-ring until $N_\beta = 2T_1+1$.

12 +1 8 (13+8=21)

13 +1 8 (14+8=22)

14 +1 8 (15+8=23)

- β-ring contains 15 nodes
- Both the α-rings contain 8 nodes each

- The ring structures will be as follows:

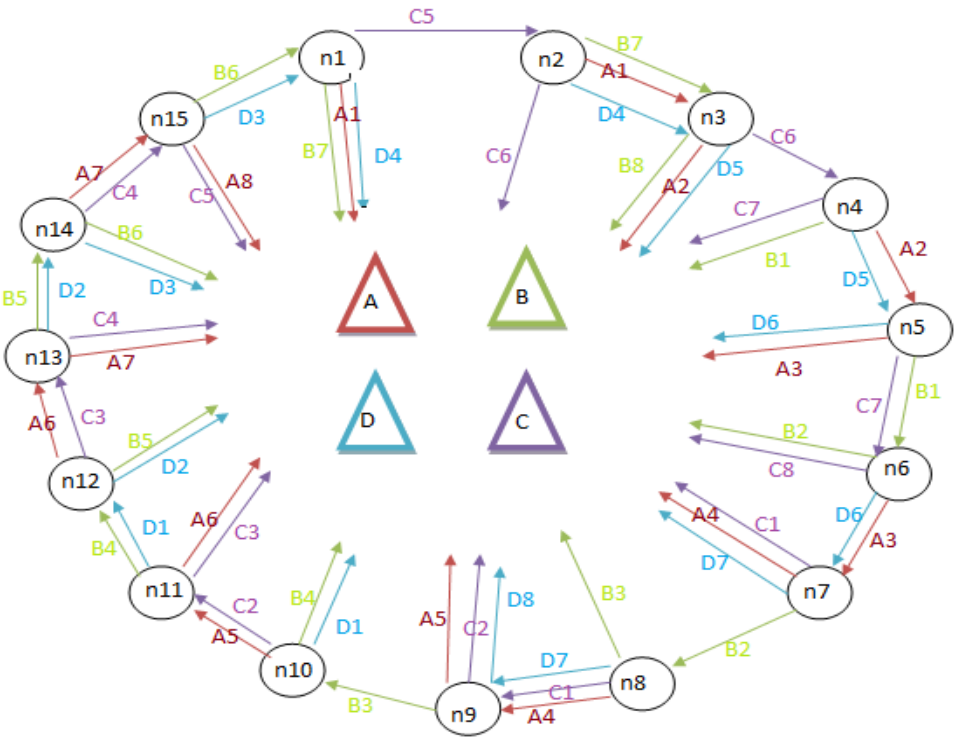

Fig. 45 β-tree with 15 nodes

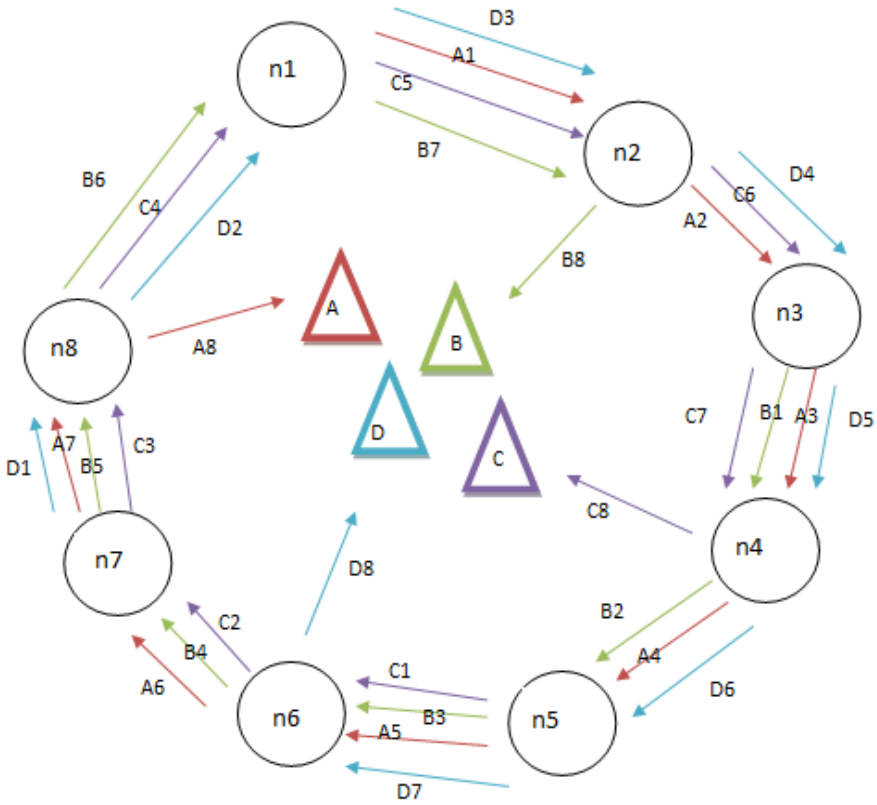

Fig. 46 α-ring with 8 nodes

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 that guarantee information collection time with least delay. This algorithm directs the issue of a faulty node. Also I developed an algorithm for fault tolerance (where a single node becomes faulty) and developed mechanism through which I can handle the fault without much overhead. In the following stage if any opportunity is available I will work upon the implementation of the algorithm.

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 inwireless 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

I) Case A - When Node becomes faulty

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

Proof:

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

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

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

for $\hat{n}$ nodes

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

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

PLAGIARSIM REPORT

161245

by A S

Submission date: 27-May-2020 07:49PM (UTC+0530)
Submission ID: 1332812897
File name: Project_Report_Final.pdf (1.6M)
Word count: 8553
Character count: 39368

161245

ORIGINALITY REPORT

17%	14%	14%	%
SIMILARITY INDEX	INTERNET SOURCES	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	3%
2	Mihaela Cardei. "Algorithms for Fault-Tolerant Topology in Heterogeneous Wireless Sensor Networks", IEEE Transactions on Parallel and Distributed Systems, 04/2008 Publication	2%
3	www.ir.juit.ac.in:8080 Internet Source	1%
4	kasanpro.com Internet Source	1%
5	machaki.me.noda.tus.ac.jp Internet Source	1%
6	www.slideshare.net Internet Source	1%
7	www.cse.fau.edu	

	Internet Source	1%
8	ira.lib.polyu.edu.hk Internet Source	1%
9	www.reaktproject.eu Internet Source	1%
10	researchbank.rmit.edu.au Internet Source	1%
11	Avinash More. "Fault-Tolerant Scheduling for Distributed Sensor Networks", 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), 2019 Publication	1%
12	ijettjournal.org Internet Source	<1%
13	dl.acm.org Internet Source	<1%
14	Gaojie Chen, Jinchuan Tang, Justin P. Coon. "Optimal Routing for Multi-Hop Social-Based D2D Communications in the Internet of Things", IEEE Internet of Things Journal, 2018 Publication	<1%
15	Cheng, Chi-Tsun, Chi K. Tse, and Francis C. M. Lau. "A Delay-Aware Data Collection Network	<1%

	Structure for Wireless Sensor Networks", IEEE Sensors Journal, 2011. Publication	
16	ijcsi.org Internet Source	<1%
17	www.computer.org Internet Source	<1%
18	Atsushi Sasaki. "A time-optimal distributed sorting algorithm on a line network", Information Processing Letters, 2002 Publication	<1%
19	jad.shahroodut.ac.ir Internet Source	<1%
20	scholarworks.alaska.edu Internet Source	<1%
21	ethesis.nitrkl.ac.in Internet Source	<1%
22	citeseerx.ist.psu.edu Internet Source	<1%
23	www.conceptsnrec.com Internet Source	<1%
24	www.ijreat.org Internet Source	<1%
25	www.ijetae.com Internet Source	<1%

26	tdl.libra.titech.ac.jp Internet Source	<1%
27	www.cityofchicago.org Internet Source	<1%
28	www.hindawi.com Internet Source	<1%
29	geb.uni-giessen.de Internet Source	<1%
30	eprints.ucm.es Internet Source	<1%
31	Sofia, Isha Batra, Vikas Verma, Arun Malik. "A comparative analysis of data collection methods in internet of things", Proceedings of the Third International Conference on Advanced Informatics for Computing Research - ICAICR '19, 2019 Publication	<1%
32	Karthikeyan, A., Rajeev Ranjan, and Kolvekar Neha Shridhar. "A novel hybrid decentralized unequal clustering approach for maximizing network lifetime in wireless sensor networks", 2013 IEEE International Conference on Computational Intelligence and Computing Research, 2013. Publication	<1%

33	hdl.handle.net Internet Source	<1%
34	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 Publication	<1%
35	cora.ucc.ie Internet Source	<1%
36	Chao Yang Lee. "Distributed Energy Harvesting Management Algorithm in Internet of Things Devices", 2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), 2019 Publication	<1%
37	M.A. Matin, M.M. Islam. "Chapter 1 Overview of Wireless Sensor Network", IntechOpen, 2012 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 14-07-2020

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

Name: ARUBA SOOD **Department:** CSE **Enrolment No:** 161245

Contact No.7018384939/9418914512 **E-mail.** arubasoo@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.

- Total No. of Pages =77 (EXCLUDING PLAGIARISM REPORT)
- Total No. of Preliminary pages =10
- Total No. of pages accommodate bibliography/references =2

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 17%. 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:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Abstract & Chapters Details</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/ Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Character Counts</td><td rowspan="2"></td></tr>
<tr><td>Report Generated on</td><td>Submission ID</td></tr>
<tr><td></td><td></td><td>Page counts</td><td></td></tr>
<tr><td></td><td></td><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete Thesis/Report in (PDF) & DOC (Word File) through your Supervisor/Guide at plagcheck.juit@gmail.com