FAULT-TOLERANCE IN CONCURRENT DATA COLLECTION TREES FOR IoT APPLICATIONS

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

in

Computer Science & Engineering

By

141203 Shobhit Kumar Srivastava
141204 Pranjal Sharma

Under the supervision of

Mr. Arvind Kumar

To

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

CERTIFICATE

This is to certify that this project report entitled ***Fault Tolerance in Concurrent Data Collection Trees for IoT Applications*** submitted to ***Jaypee University of Information Technology***, is a bonafide record of work done by

141203 Shobhit Kumar Srivastava

141204 Pranjal Sharma

under my supervision from ***January, 2018*** to ***May, 2018*** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology*** *in* ***Computer Science & Engineering***.

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

Mr. Arvind Kumar
Assistant Professor
Department of Computer Science & Engineering and IT
Dated:

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Project Supervisor Mr. Arvind Kumar as well as our Project Coordinator Dr. Punit Gupta who gave us the golden opportunity to do this wonderful project on the topic ***Fault Tolerance in Concurrent Data Collection Trees for IoT Applications****, which also helped us in doing a lot of Research and we came to know about so many new things we are really thankful to them. Secondly we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.*

(Pranjal Sharma) *(Shobhit K Srivastava)*

TABLE OF CONTENT

LIST OF 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
MCU	Microcontroller Unit
TCR	Transceiver Unit
SB	Sensor Board

E_{i_SN}	Energy consumed by Sensor board
E_{i_MCU}	Energy consumed by Microcontroller Unit
E_{i_TCR}	Energy consumed by Transceiver Unit

LIST OF FIGURES

LIST OF GRAPHS

ABSTRACT

Wireless sensor nodes are prone to failure in hostile environments therefore, in this paper, a fault tolerant algorithm is described for a wireless sensor network - Concurrent Data Collection Trees. As the network becomes faulty it should be restored to its normal working within an effective time-frame. But in case of Concurrent Data Collection trees, the complexity to reconstruct the network increases as the number of parallel streams increase. Concurrent Data Collection Trees offer effective data collection and in some cases the data is not lost completely if a fault occurs. We have discussed various cases which shed light on the data that is either completely lost or not. Two different network structures namely alpha and beta rings along with their advantages and disadvantages is discussed.

Chapter I

INTRODUCTION

IoT networks are gaining popularity because of their widespread use in day-to-day activities and their promising future applications. To support urbanization in coming years, it is significant for the natives to get up and coming data in a convenient way by embracing current advances in technology. Internet of Things (IoT) is a better solution among these latest technologies. Multiple users will own and share the IoT systems, an arrangement of sensors and middleware. Various inquiries will be put together by clients and even IoT gadgets in the meantime, which initiates different parallel information streams in same system. Sensors nodes are battery powers devices. Energy saving is very significant to lifetime of a sensor network. The amount of energy consumed in a transmission is proportional to the corresponding communication distance. Therefore, long distance communication between nodes and base station are usually not encouraged. The consumption of energy 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 communicates. These sub-cluster heads in turn are connected to the cluster head which controls all clusters and finally 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".

Wireless sensor networks are used in a variety of applications; therefore they possess a number of characteristics which are crucial for their successful working. There have been many types of networks around for wireless sensors but considering Cheng's et. al. Concurrent Data Collection Trees, can improve data collection efficiency as they use parallel data streams for data transmission. But energy saving becomes crucial to the aspect of battery powered devices. In IoT networks there are a number of devices operating together, interconnected to each other and require a lot of energy while operating. Therefore a network has to be constructed in such a way that it uses less power while at the same time has less delay in transmitting data. 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 limitation is fault in networks where three cases have been discussed – node becomes faulty, base station becomes faulty, and both become faulty. But in case of parallel networks and as complex as Concurrent Data Collection Trees the task of making them fault tolerant becomes difficult. Therefore, in this project 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 discussed different cases as to which network structure is better and the different possibilities of its restoration at different time slots associated with it. Our, results have been verified using simulations carried out using the code we developed in Python.

RELATED WORK

The problem of fault tolerance has been discussed previously by many authors but a parallel network such as Concurrent Data Collection Tree has not received much attention. Although parallel network structures have been discussed before but Concurrent Data Collection Tree is much more flexible and efficient in terms of data transfer and processing. In [3] the author has discussed about the fault occurring in Wireless sensor grid networks. The author has discussed about the levelling algorithm which is energy efficient. The transmission range method is described which better than levelling algorithm in terms of a network that is partitioned. The two algorithms are then combined which minimize the chances of network partitioning. In [4] an algorithm is proposed for a network possessing a tree topology. The algorithm proposed is a distributed sorting algorithm which takes advantage of Dijkstra's model in terms that the fault need not be corrected manually but rather is solved by the software architecture described in the algorithm. In [5] the fault-tolerant described is used for controlling fault in heterogeneous wireless sensor networks. The k-degree Anycast Topology Control (k-ATC) problem is discussed along with its three solutions: the greedy centralized algorithm for minimizing the transmission range between all wireless sensor nodes and a distributed and localized algorithm for adjusting the wireless sensor nodes' transmission range. The algorithm described here also minimizes energy consumption. In [6] the author has presented two algorithms: Full 2-Connectivity Restoration Algorithm (F2CRA) and the Partial 3-Connectivity Restoration Algorithm (P3CRA) for fault tolerance. F2CRA and P3CRA construct fan shaped topology and dual ring topology to restore the network to its normal working. F2CRA and P3CRA are suitable in scenarios where cost and performance are given prime importance. In [7] the author has discussed about the issues of fault detection and its data recovery. To solve such problems a cluster based technique is suggested using genetic algorithm. Clustering the network according to Energy-Efficient Distance-Based Clustering Algorithm and using GA some nodes are selected as backup nodes. Such clustering algorithm helps in detecting the faults occurring in nodes. In [8] the author has discussed about the effect of re-clustering the network after the

fault has occurred, he told when the network is reconstructed when cluster head is faulty there is always a loss of energy and time. In order to remove these loss he proposed the genetic algorithm which take into consideration three parameters in clustered network which are distance between every node from cluster heads, remained energy of cluster heads and member number of each cluster head. Result of simulation shows that this genetic algorithm has better result of recovering cluster head than that which consider only one parameter. In [9] the author has presented a new method for fault tolerance and detection. This method is based on majority vote, which result in detecting the faulty nodes accurately. The researchers compared this genetic algorithm with Chen, Lee, and hybrid algorithms. Result shows that this genetic algorithm based on majority vote method his more efficient in detecting false alarm rate and detection accuracy.

PROJECT OBJECTIVE

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

Chapter II

LITERATURE SURVEY (I)

"A Delay-Aware Data Collection Network Structure for Wireless Sensor Networks"

Abstract

In any network to save energy is very significant. Here in this paper i.e. "Delay aware data collection network structure for wireless sensor networks is proposed", two algorithms are suggested and implemented to deal with this very problem of minimizing energy to gain longer life time, for that even the efficiency of collecting data is also ignored.

Top-Down Approach

1. Begin with a mess network i.e. fully connected network with nodes N >= 4.

Considering N=8

All the nodes are with connection degree=7

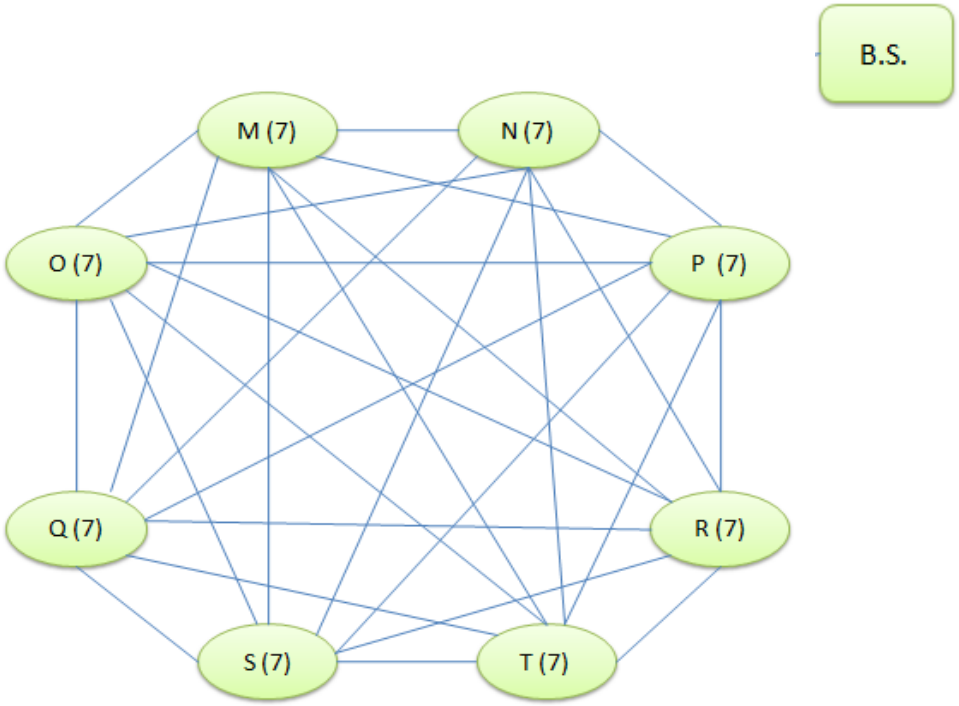


Fig 1(a). Initial Network Structure

These 8 nodes will form $\widehat{H}_s$, where s=1 i.e. $\widehat{H}_1$

Now define parameter b=N/2 = 4

1. Select b=4 nodes from $\widehat{H}_1$ to form H_2, such that the total edge weight within set H_2 is maximized.

   As all have same edge weight, taking {O,P,E,R} to form H_2

   $\widehat{H}_2$ = {O, P, Q, R}

   H_2 = {M, N, S, T}

   Cut all the connections among nodes in H_2

   Set b=b/2=2

   Set s=s+1 = 2

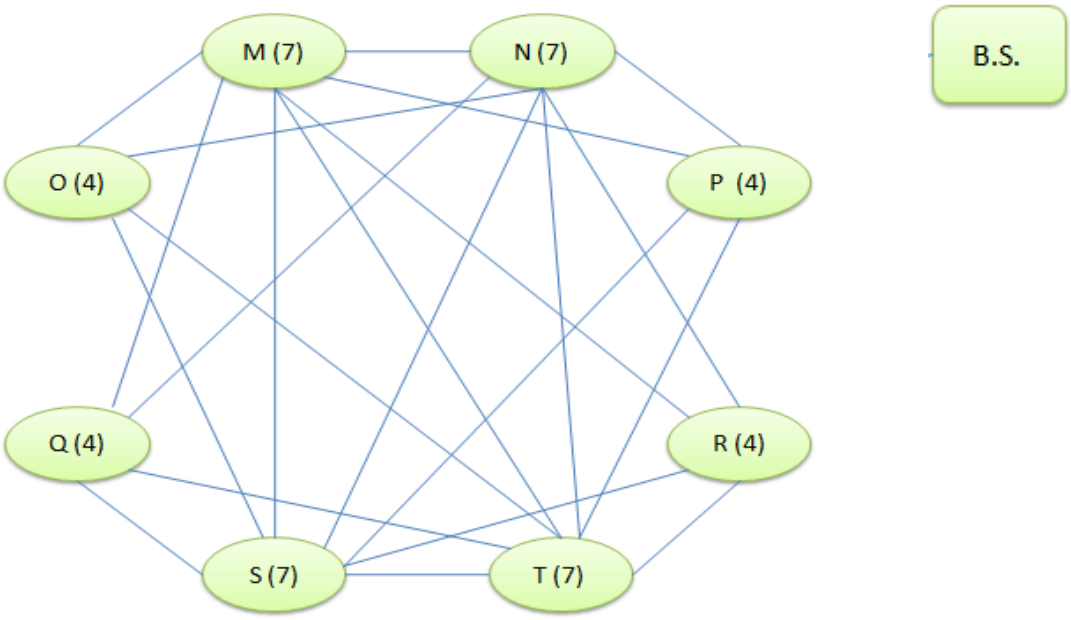


Fig 1(b). Network Structure after iteration 1

2. Since b=2 >=2, the previous step is repeated

   Select b=2 from set $\widehat{H}_2$ to form set H_3

   H_3 = {M, T}

   $\widehat{H}_3$ = {N, S}

   Cut all the connections between nodes in set H_3

   Set b=b/2 = 1

   Set s=s+1=3

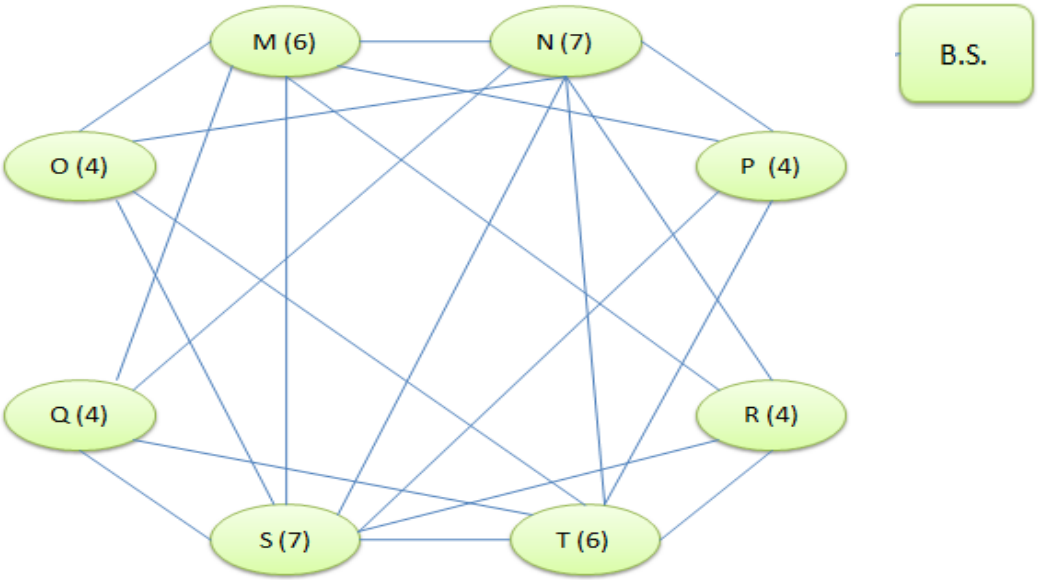


Fig 1(c). Network Structure after iteration 2

3. With b=1<2 , Algo proceeds and define parameter r=2

   Nodes with degree=N-r =6 (i.e. M, T) forms L

   Nodes with degree>N-r (i.e. N, S) forms U

   Reduce connections among nodes between set L & U until each node in set L is only connected to a single node in set U, provided that total edge weight is minimized.

   Set r=2r=4

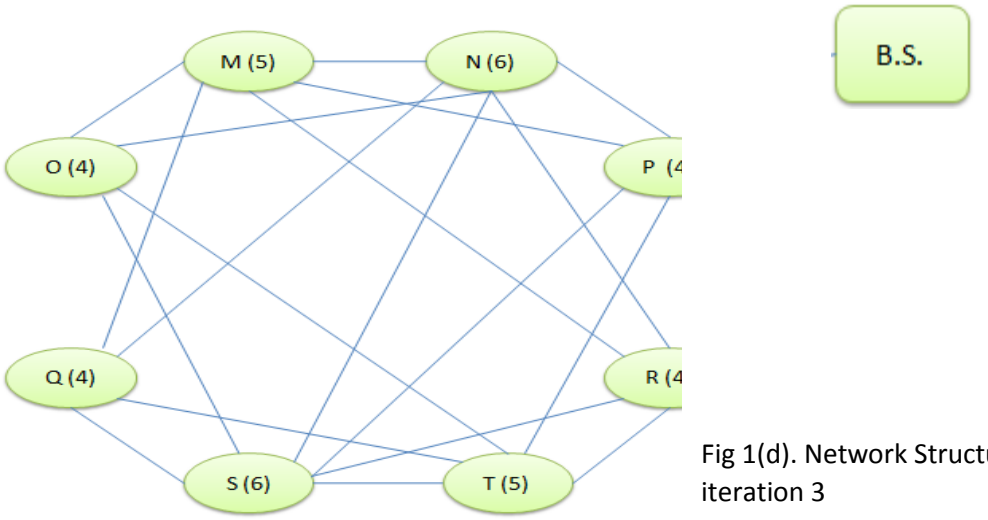


Fig 1(d). Network Structure after iteration 3

4. Since r=4<N, previous step is repeated.

   Nodes with degree = N-r = 4 (i.e. O, P, Q, R) forms L

   Nodes with degree > N-r (i.e. M, N, S, T) forms U

   Reduce connections among nodes between set L & U until each node in set L is only connected to single node in set U

   Set r=2r=8

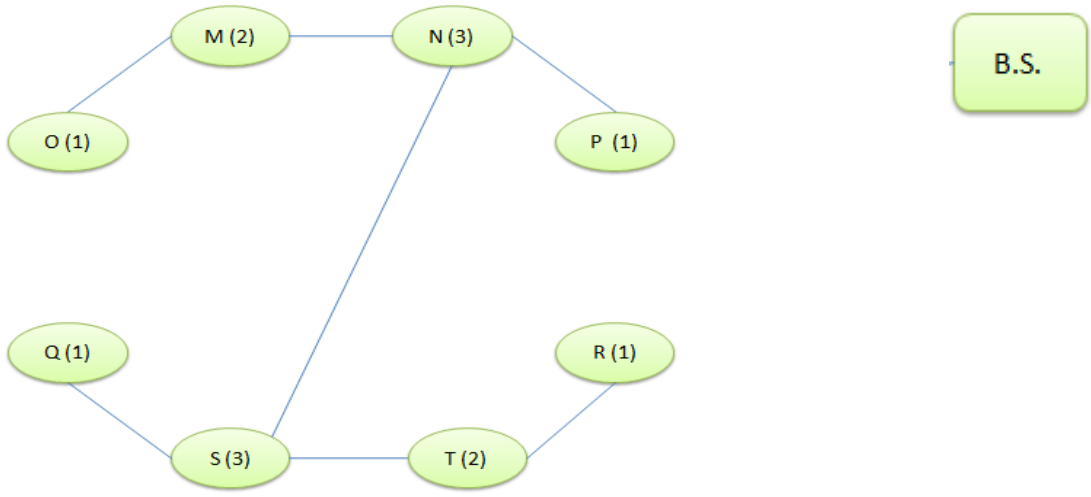


Fig 1(e). Network Structure after iteration 4

5. When r=8>N, the basic operation of proposed top down approach is completed. Resultant network will now consist of two nodes with degree $\log_2(N)=3$ (i.e. N & S) The one closer to B.S. will be selected as cluster head and be directly connected to base station.

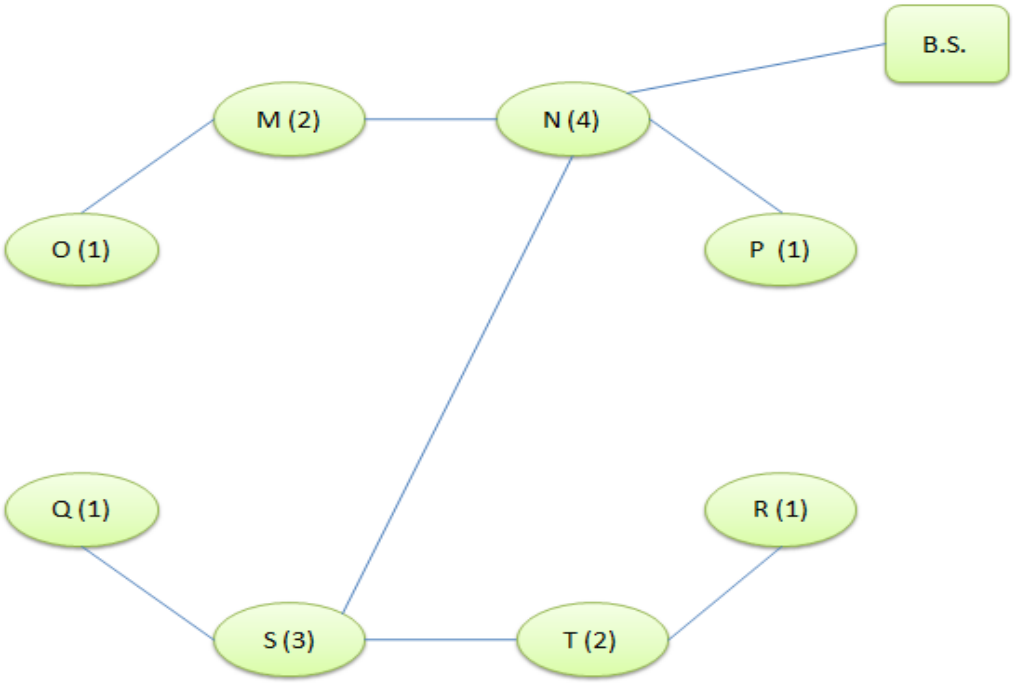
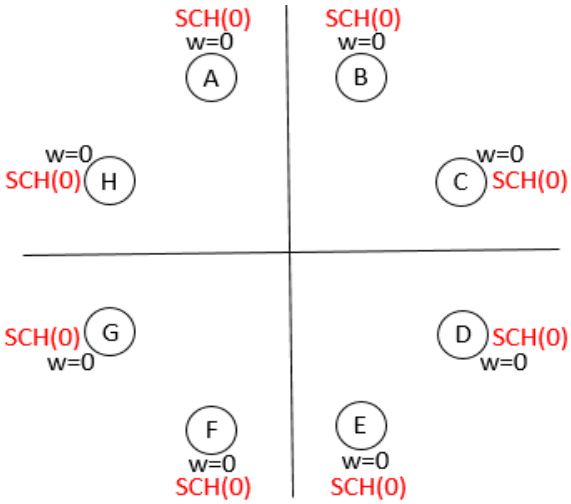


Fig 1(f). Final Network Structure

Bottom-Up Approach

1. Step 1



Node A:
(-1, 2) and (7, 7)

$$r_{dp} = \sqrt{t_x^2 + t_y^2}$$

$$r_{dp} = 9.43\ m$$

$$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1$$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 2.35\ m$$

Fig 2(a). Initial Network Structure

2. Step 2

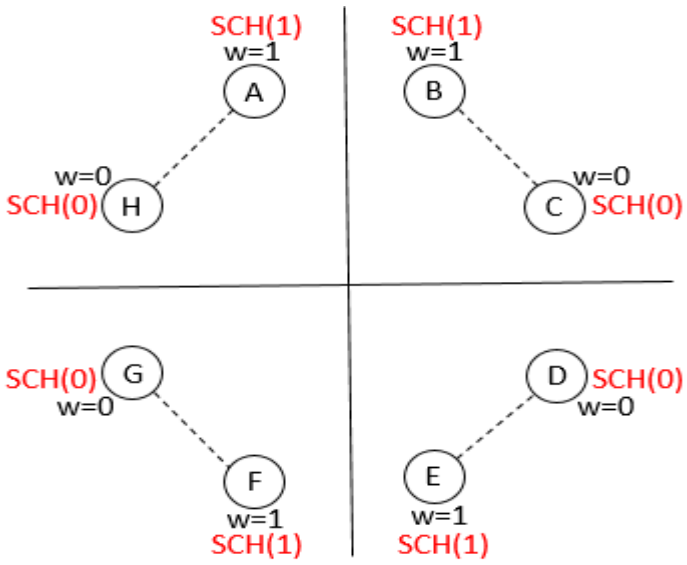


Fig 2(b). Network Structure after iteration 1

Node A:
(-1, 2) and (7, 7)

$$r_{dp} = \sqrt{t_x^2 + t_y^2}$$

$$r_{dp} = 9.43\ m$$

$$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1$$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 2.35\ m$$

Node A:
(-1, 2) and (7, 7)

$$r_{dp} = 9.43\ m$$

$$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1; \omega = 1$$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 3.14\ m$$

3. Step 3

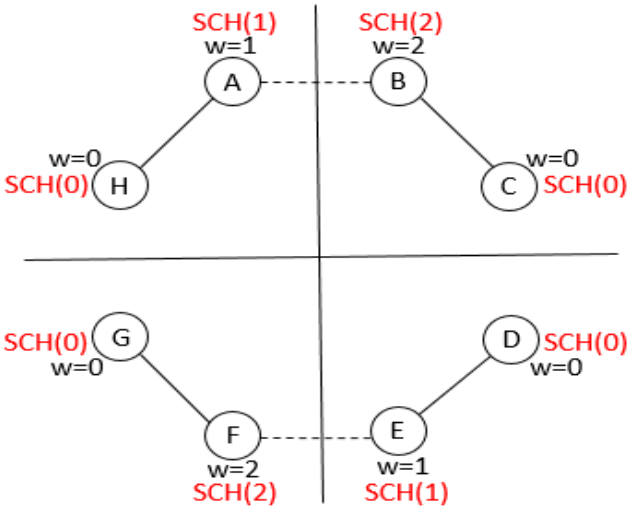


Fig 2(c). Network Structure after iteration 2

Node A:
(-1, 2) and (7, 7)

$r_{dp} = 9.43\ m$

$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1; \omega = 1$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 3.14\ m$$

Node B:

$r_{dp} = 9.43\ m$

$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1; \omega = 2$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 4.71\ m$$

4. Step 4

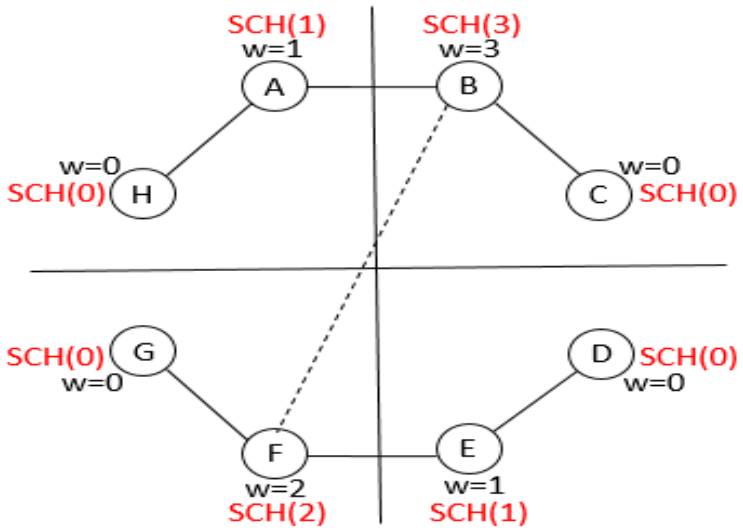


Fig 2(d). Network Structure after iteration 3

Node B:

$r_{dp} = 9.43\ m$

$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1; \omega = 2$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 4.71\ m$$

Node B:

$r_{dp} = 9.43\ m$

$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1; \omega = 3$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 9.43\ m$$

$r_{com} = r_{dp}$

$\therefore$ ***Node B makes connection with base station.***

5. Step 5

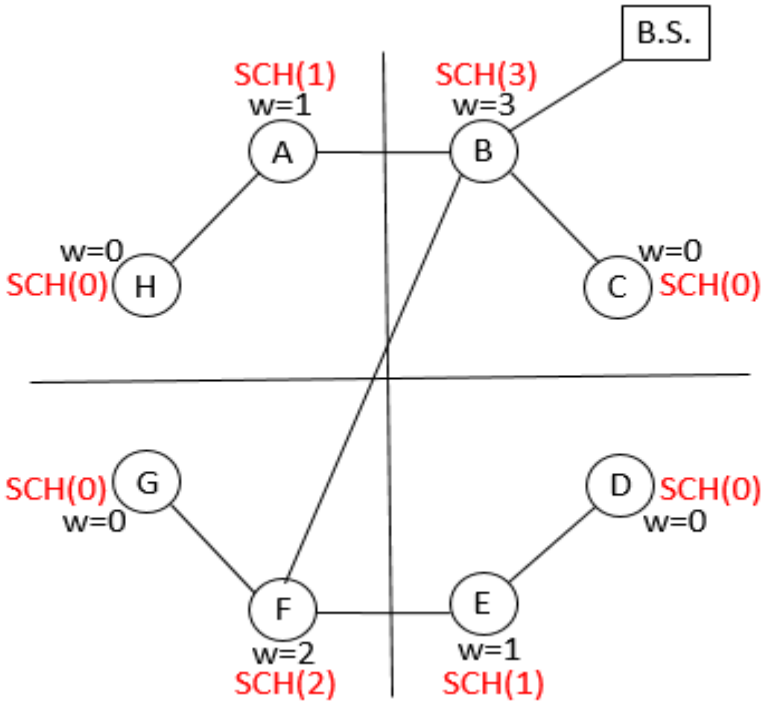


Fig. 2(e) Final Structure of Network made with Bottom-Up Approach

Node B:

$$r_{dp} = 9.43\ m$$

$$\alpha = 4; \alpha = \lceil \log_2 N_{est} \rceil + 1; \omega = 3$$

$$r_{com} = \frac{\sqrt{t_x^2 + t_y^2}}{\alpha - \beta - \omega} = 9.43\ m$$

$$r_{com} = r_{dp}$$

$$\therefore \textit{Node B makes connection with base station.}$$

Performance Analysis (I)

Type 1 – Single Cluster Network

I. Data Collection Time – Lower the better

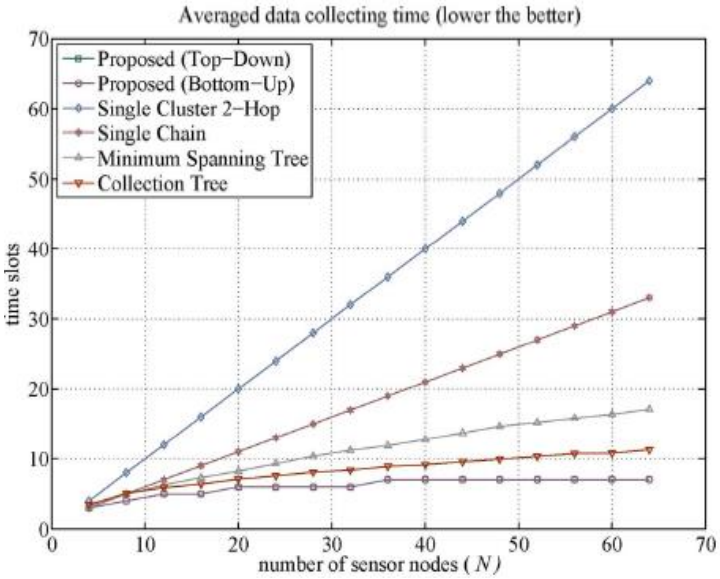


Fig. 3 Averaged Data Collecting Time

1. Proposed network structure (DADCNS).
2. Collection Tree Protocol.
3. Minimum Spanning Tree Protocol.
4. Single Chain.
5. Single Cluster 2 Hop (SC2H).

II. Minimizing ψ- Lower the better

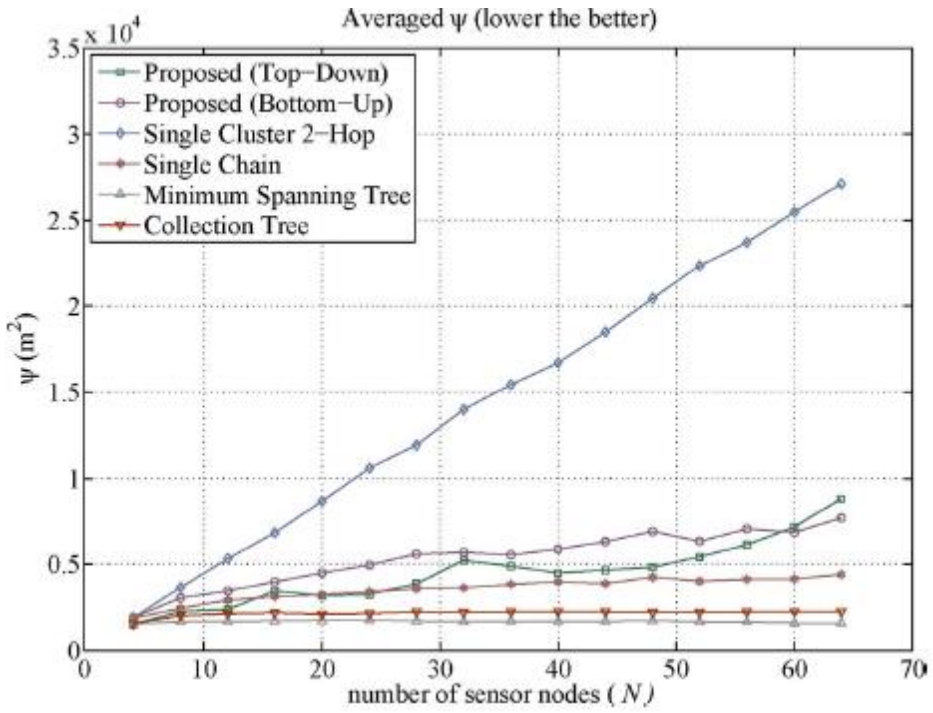


Fig. 4 Averaged ψ

1. Minimum Spanning Tree Protocol.
2. Control Tree Protocol.
3. Single Chain.
4. Top-down approach (DADCNS).
5. Bottom-up approach (DADCNS).
6. Single Cluster 2 Hop (SC2H).

III. Network Lifetime – Higher the better

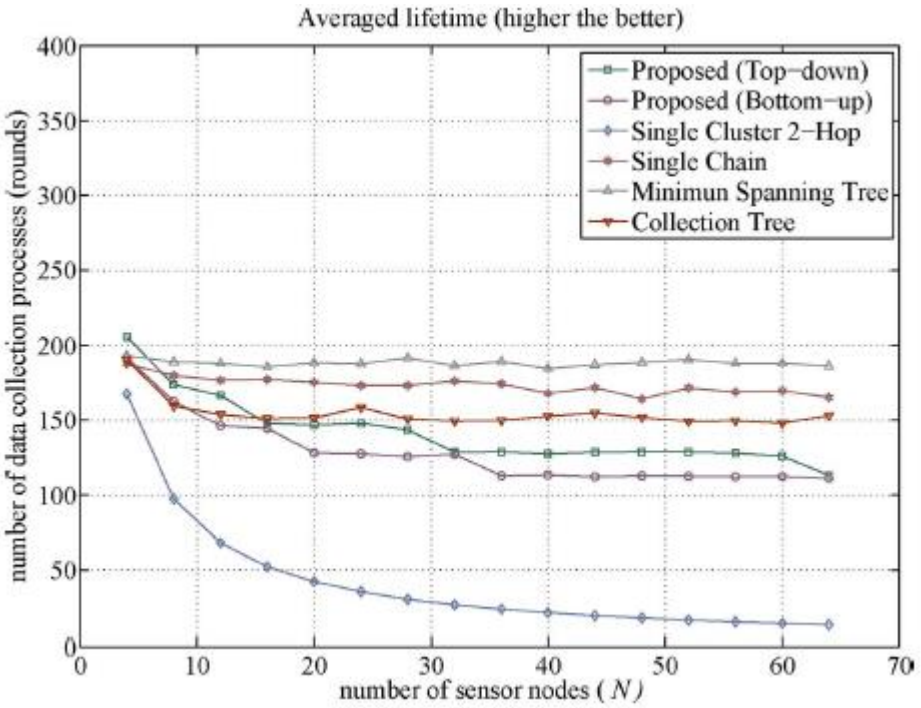


Fig. 5 Averaged Lifetime

1. Minimum Spanning Tree Protocol for N > 4
2. Single Chain.
3. Control Tree Protocol.
4. Top-down approach (DADCNS).
5. Bottom-up approach (DADCNS).
6. Single Cluster 2 Hop (SC2H).

Type II – Multiple Cluster Network

I. Data Collection Time – Lower the better

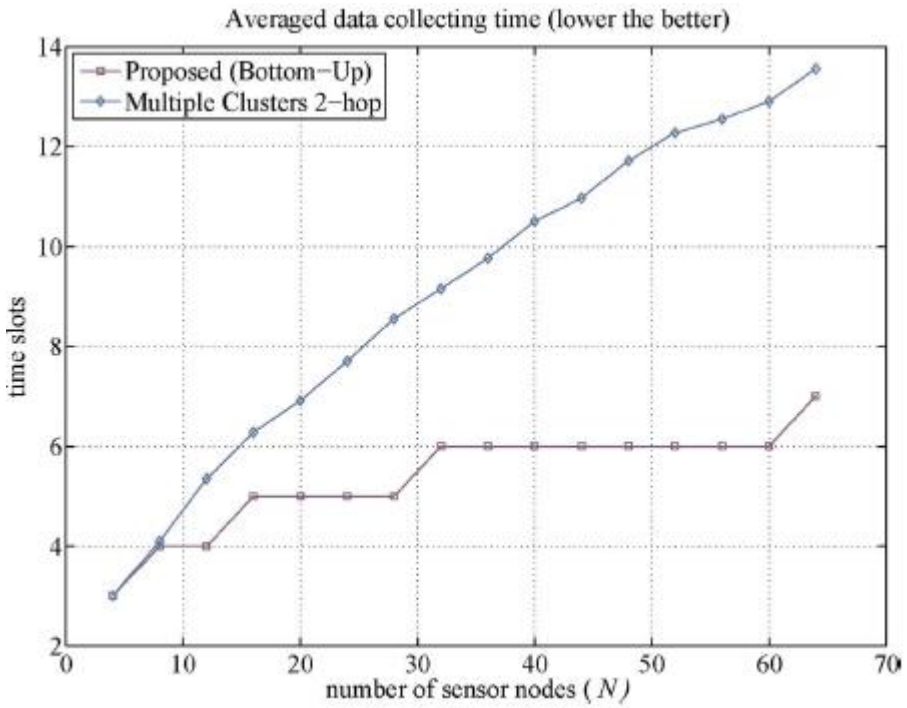


Fig. 6 Averaged Data Collecting Time

1. Proposed Network Structure (DADCNS).
2. Multiple Cluster 2 Hop (MC2H) – DCT increases as N increases.

II. Minimizing ψ - Lower the better

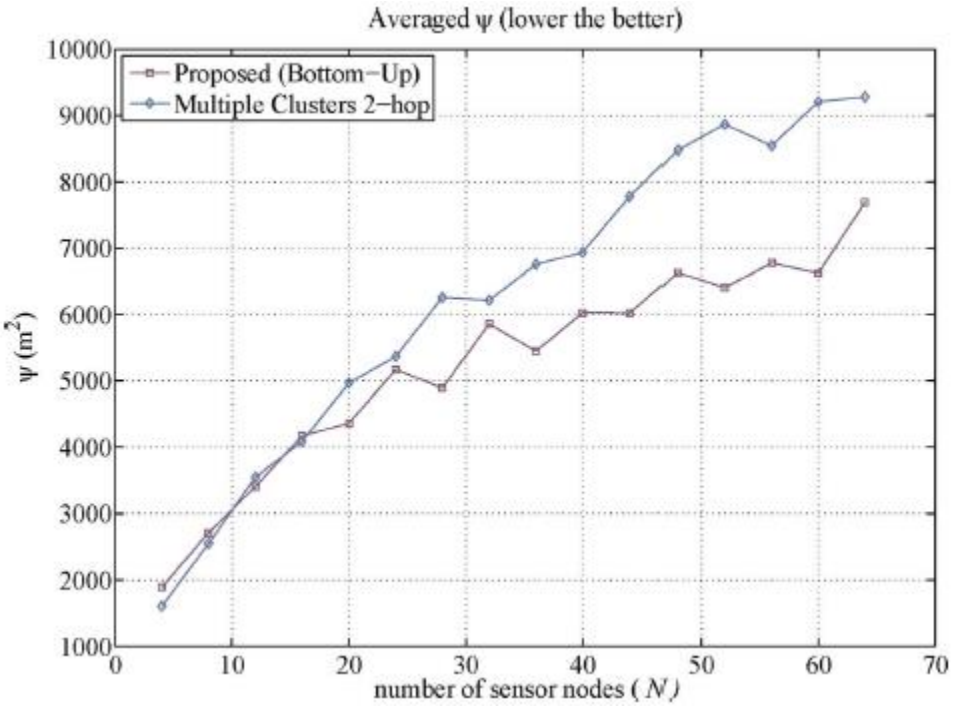


Fig. 7 Averaged Ψ

1. Both give similar performance for $N \leq 12$
2. $N > 12$
    a. Proposed Network Structure (DADCNS).
    b. Multiple Cluster 2 Hop (MC2H).

III. Network Lifetime – Higher the better

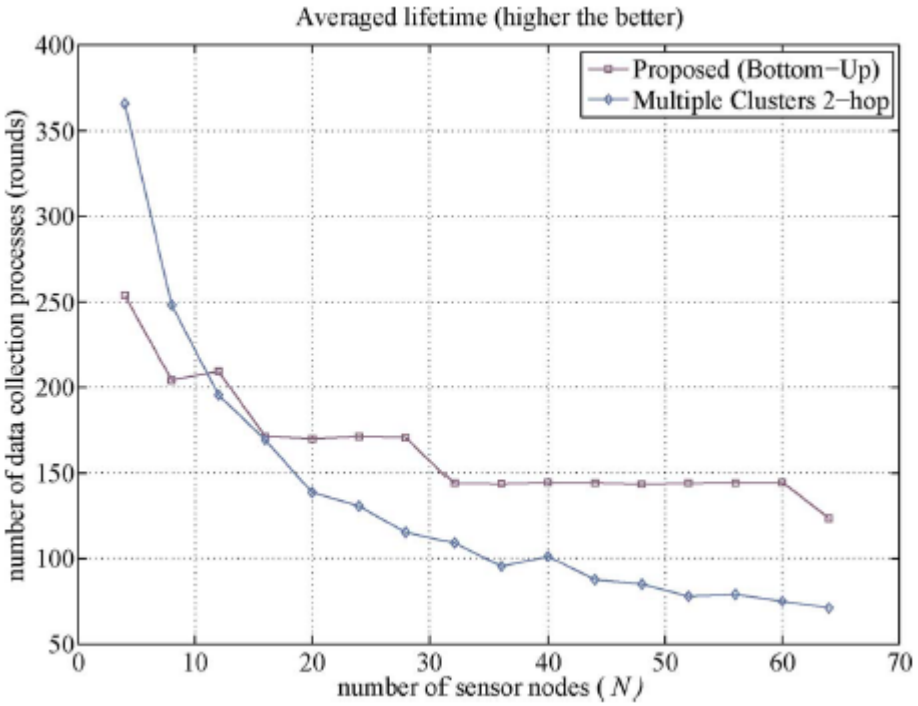


Fig. 8 Averaged Lifetime

1. Proposed Network Structure (DADCNS).
2. Multiple Cluster 2 Hop (MC2H).

LITERATURE SURVEY (II)

"Concurrent Data Collection Trees for IoT Applications"

Abstract

In IoT network a lot of devices work together. They generate massive amounts of data thus data collection process becomes a chief concern in large networks. Data collection processes must utilize minimum amount of time while collecting data. For this reason a new network structure – Concurrent Data Collection Trees for IoT networks has been proposed in this paper. This network minimizes data collection time to an acceptable value. Another concern is that if a large network, as large as IoT network encounters any fault. Base Station or a node in a network might become faulty because of many reasons – environmental conditions, battery run out, improper handling, etc. To solve this problem we are trying to design an approach for fault tolerance in concurrent data collection tree network structure.

Concurrent Data Collection Tree

It is a network structure N = {n_1, n_2, n_3, ……,$n_{|N|}$} nodes and S = {s_1, s_2, s_3, ….., $s_{|S|}$} base stations assuming that all IoT nodes communicate with each other and base station. Data fusion technique fuses multiple packets of data (from IoT nodes) into one single packet before it is forwarded to one's parent node. For concurrent data collection processes it must employ equal number of data streams and base stations – "Each concurrent data aggregation process will use a different base station (BS) to access the IoT network and the total number of concurrent data streams is *k*". Concurrent data streams must use equal number of nodes determined by equation (1). This equation determines maximum number of streams a node must utilize so that data collection time is minimum. This networks structure has a constraint: $|N| \geq k$.

$$u_{max} = \left\lfloor \frac{|N|}{k} \right\rfloor \qquad (1)$$

The following equation represents number of hubs used by an information stream in i[th] vacancy.

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

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

$$\hat{u}_j = \left\lceil \frac{u_j}{2} \right\rceil (3)$$

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 T_1 time slot all nodes and base stations communicate in concurrent fashion, but in T_2 time slot the left over nodes communicate using DADCNS. The leftover nodes are calculated using $|N| - \tau_1 \left\lceil \frac{u_{max}}{2} \right\rceil$.

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

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

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

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 valid only for $u_{max} = 2$. "A data stream in an α-ring N_α will need T_1 time-slots to aggregate data from $|N_\alpha|$ nodes onto a single node. Such anode will take one time-slot to report the fused data to the BS". If nodes are numbered arbitrarily then at any time slot two nodes that will communicate (during K^{th} data collection process) i.e. node n_{c1} transmits data to node n_{c2}.

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

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

Data fusion will be performed on node n_{c2}. Overall duration can be calculated using T_1 and T_2 from equations (2) &(3).

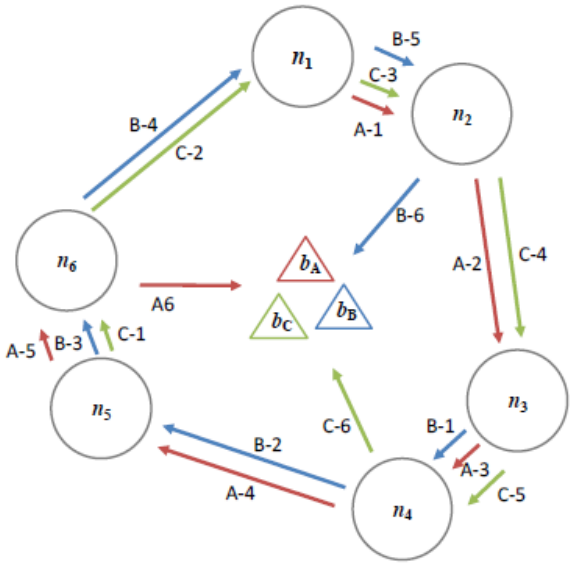


Fig. 9 Alpha ring with 6 nodes

$T = T_1 + T_2 = 5 + 1 = 6$.

B) The β-ring

For $u_{max} = 3$ and $|N_\beta| \geq 3k$ the network is known as β-ring. 2 nodes will communicate using node-to-node (N2N) communication while the other node will communicate using node-to-base (N2BS) communication. If nodes are numbered arbitrarily then at a particular time slot of K^{th} data collection process node n_{c3} will be involved in node-to-base station communication and n_{c4} will transmit data n_{c5}.

c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|))

c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|))

c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|))

Data fusion will be performed on node nc5.

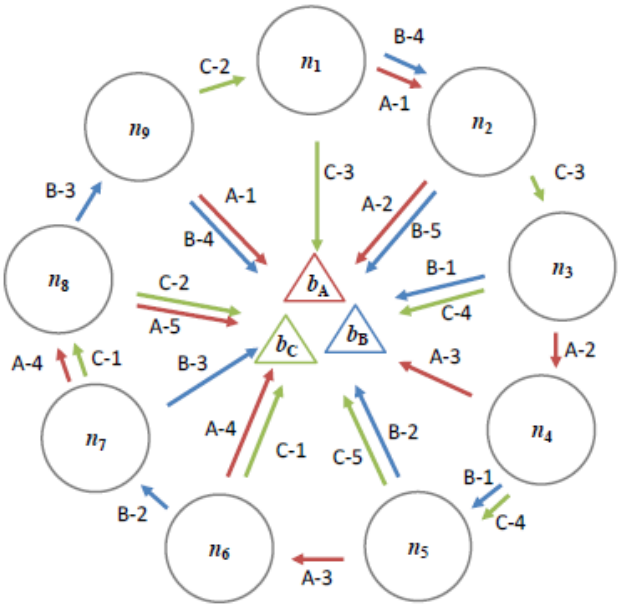


Fig. 10 Beta ring 9 nodes

$T = T_1 + T_2 = 4 + 1 = 5.$

C) Multiple Rings

I) **$u_{max} \geq 4$**: $n_\alpha = \frac{u_{max}}{2}$ α-rings are formed. Each of the α-rings formed will first be granted 2*k* nodes. The rest |N|- $n_\alpha(2k)$ nodes will then be granted to those n_α rings one by one. Each α-ring will operate independently as described above.

II) **$u_{max} \geq 5$**: A single β-ring with $n'_\alpha = \frac{u_{max}-3}{2}$ number of α-rings are formed. Initially, β-ring will be granted 3*k* nodes, while each α-ring will be granted 2*k* nodes. The rest $|N|-3k_{BS} - n'_\alpha(2k_{BS})$ node will be granted to the β-ring until $|N_\beta| = 2T_1 + 1$. The remaining nodes will be distributed to α-rings one by one. An algorithm for constructing such a network structure is described in the Algorithms section.

Results & Analysis

The duration of data collection process T (total number of spaces required by base stations of different data streams) is used as a performance indicator. The reference structure used 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 comparatively low in comparison to DADCNS.

- As value of k increases the value of T also increases linearly in DADCNS while in the concurrent data collection tree structure with increase in value of |N|, value of u_{max} also increases but the total value of T increases slightly. Therefore with increase in value of |N| & k the performance gap between two network structures also widens.

- When value of k or |N| is increased, value of u_{max} also changes. This change in the value of u_{max} leads to variations in the number of α and β rings. Therefore it is observed that when k or |N| increases the value of T does not increase monotonically.

Chapter III

ALGORITHMS

A) Bottom-Up Approach (I)

- Initially all nodes are disconnected and each node is represented as SCH(0) – Sub cluster head of level 0.
- Each SCH performs random back-off and transmits density probing packet to its neighboring SCHs within r_{dp} distance.
- Each SCH will do a random back off and then broadcast an *invitation packet* (IVP) to its neighbors within r_{com} distance.
- SCHs receiving IVP will send connection request and nearest distance neighbors will form a connection belonging to level w+1 clusters.
- Distance r_{com} will be increased if no connection can be formed within a specified time frame and the SCH will transmit CR again at this new r_{com}.
- When $r_{com} = r_{dp}$ the SCH of the cluster will form connection with the base station.
- The above process continues until no more connection can be formed.

B) Fault Tolerance Algorithm for Concurrent Data Collection Tree

- **Case A – When Node becomes faulty**
  - If the node does not acknowledges its neighboring node within a specific time frame an acknowledgement packet is resent.
  - Again if the acknowledgement is not received within that time frame, there is a high probability that the node is faulty.
  - Calculate $u_{max} = \left\lfloor \frac{|N|-\hat{n}}{k} \right\rfloor$.
  - Calculate the new values of τ_1& τ_2.
  - Reconstruct the network structure with new value of c_1, c_2, c_3, c_4 and c_5.

- **Case B – When Base Station becomes faulty**
  - If the base station does not acknowledges the node within a specific time frame an acknowledgement packet is resent.
  - Again if the acknowledgement is not received within that time frame, there is a high probability that the base station is faulty.
  - Calculate $u_{max} = \left\lfloor \frac{|N|}{k-\hat{b}} \right\rfloor$.
  - Calculate the new values of τ_1& τ_2.
  - Reconstruct the network structure with new value of c_1, c_2, c_3, c_4 and c_5.

- **Case C – When both node and base station are faulty**
  - If the base station or node does not acknowledges the node or its neighboring node within a specific time frame an acknowledgement packet is resent.
  - Again if the acknowledgement is not received within that time frame, there is a high probability that the base station or node is faulty.
  - Calculate $u_{max} = \left\lfloor \frac{|N|-\hat{n}}{k-\hat{b}} \right\rfloor$.
  - Calculate the new values of τ_1& τ_2.
  - Reconstruct the network structure with new value of c_1, c_2, c_3, c_4 and c_5.

Chapter IV

PERFORMANCE ANALYSIS

Below a set of simulations are performed for each step which depicts different cases in alpha and beta ring scenario and as to what extent data can be recovered in case of a fault. Based on these simulations theoretically here and in code implementation in python we have been able to discover different properties of alpha and beta rings. In the three cases where a base station becomes faulty, node becomes faulty and both become faulty only the network structure changes while their working remains almost same with same properties applied to each network structure. Below a comparison of properties of both alpha and beta rings is provided which each serves as a different case in different scenario of a faulty network and how it can be recovered.

Analysis:

A. Base Station Faulty:

a) Alpha Ring:

In case of an alpha ring the data can be recovered completely if the fault occurs just before the last time slot during which the data is transferred to the base station. The data aggregation nodes contain the data of all the nodes and if a fault occurs, can be used to recover the data. However, all other nodes that do not take part in the data aggregation process do not contain data only of the node just next to them and hence are not that much effective in data recovery. In another scenario if the fault occurs during some intermediate value of the time slot, then the whole network needs to be reconstructed again.

b) Beta Ring:

In this network the data loss is significantly more in comparison to alpha ring as some nodes have already transferred data to the base station and during failure even the data aggregation nodes are not able to fully recover the data as they are loosely connected because of their network communication structure.

B. Node Faulty:

a) Alpha Ring:

Suppose as given in the figure below the fault occurs during some initial value of the time slot then there is either little or no loss at all and the network can be reconstructed again. This results in less time complexity since the network where the fault occurs is reconstructed immediately thereby restoring its normal functioning. However, in the intermediate time slots, a fault may result in significant loss of data with it requiting to reconstruct it from the start. The concluding time slots scenario works similar to the "Base Station Faulty" situation.

b) Beta Ring:

The beta ring in this scenario work similar to the "Base Station Faulty" situation.

C. Node and Base Station Faulty:

a) Alpha Ring:

The alpha ring can behave in accordance with the "Base Station Faulty" or "Node Faulty" depending on the situation as shown below in the figure.

b) Beta Ring:

There is no exception for beta ring in this situation and will behave according to the "Base Station Faulty" situation.

CASE I: BASE STATION FAULTY

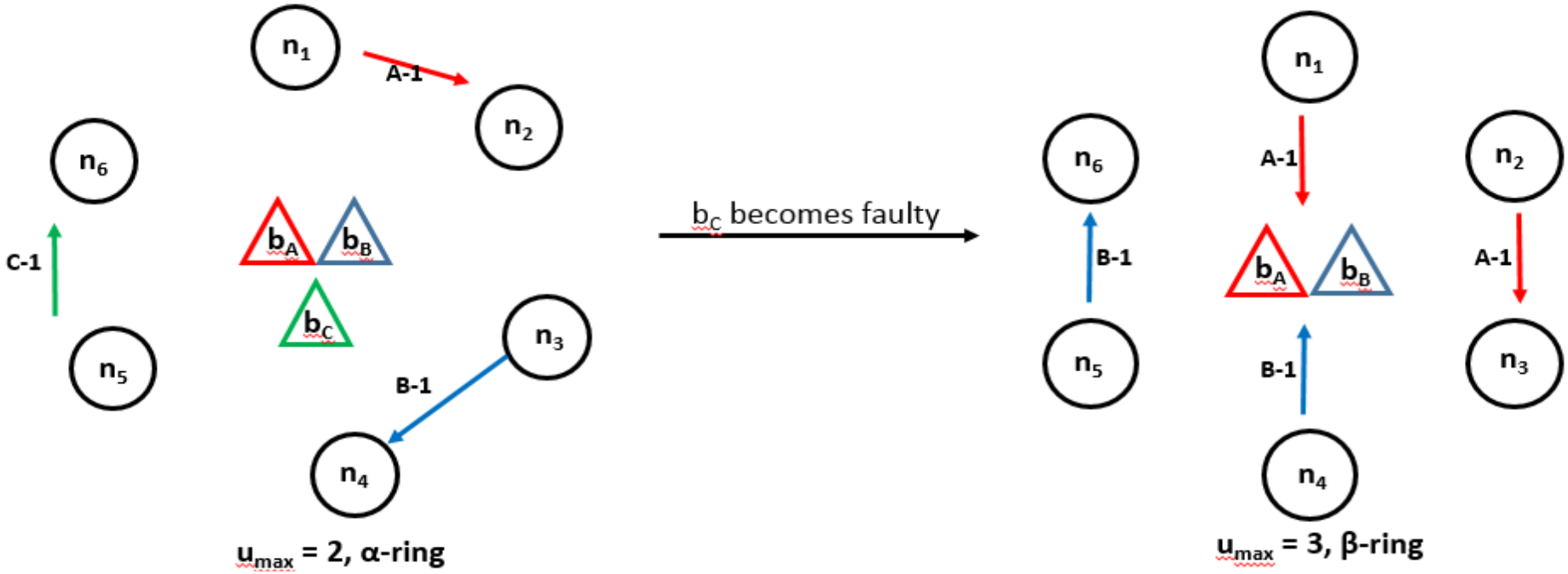


Fig. 11 Initial Alpha Ring with 6 nodes Ist time slot
K=1, 2, 3. ***t = 1.*** **N_α = 6**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 1, 3, 5
c_2 = (1 + mod(2(*K*-1) + t, |N_α|)) = 2, 4, 6

Fig. 12 Initial Beta Ring with 6 nodes Ist time slot
K=1, 2. ***t = 1.*** **N_β = 6**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) = 1, 4
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = 2, 5
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) = 3, 6

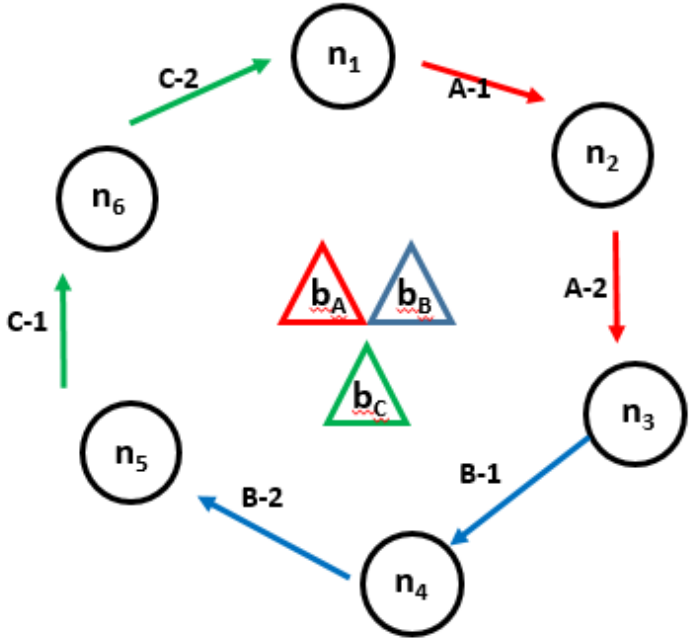


Fig. 13 Alpha Ring with 6 nodes IInd time slot
K=1, 2, 3. ***t = 2.*** **$N_\alpha = 6$**
$c_1 = 2, 4, 6$
$c_2 = 3, 5, 1$

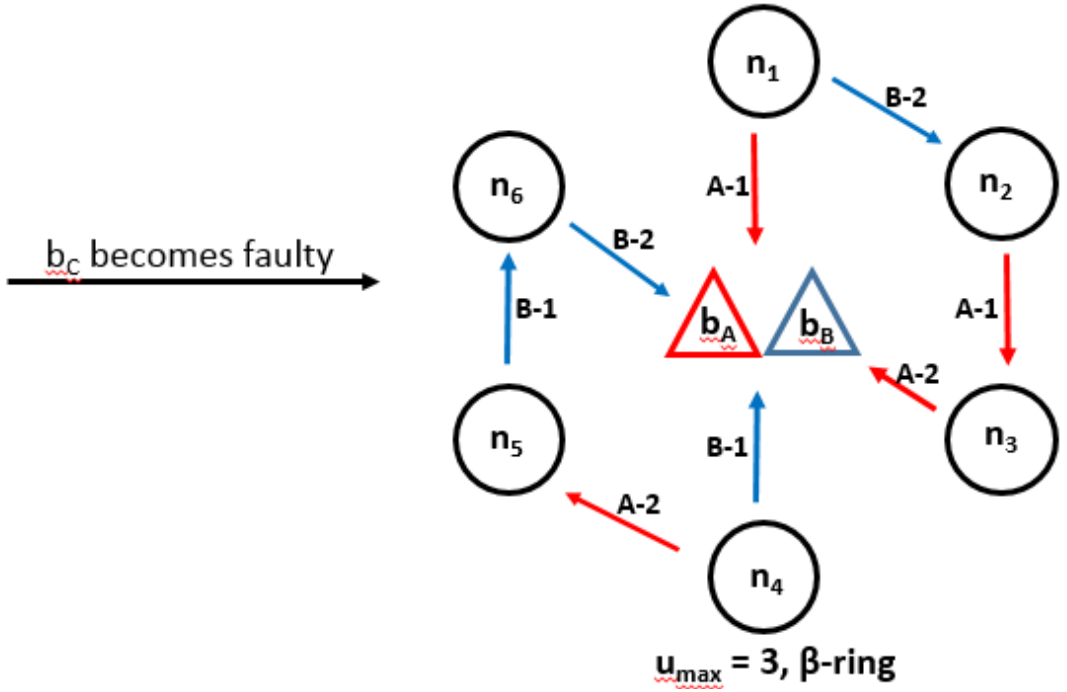
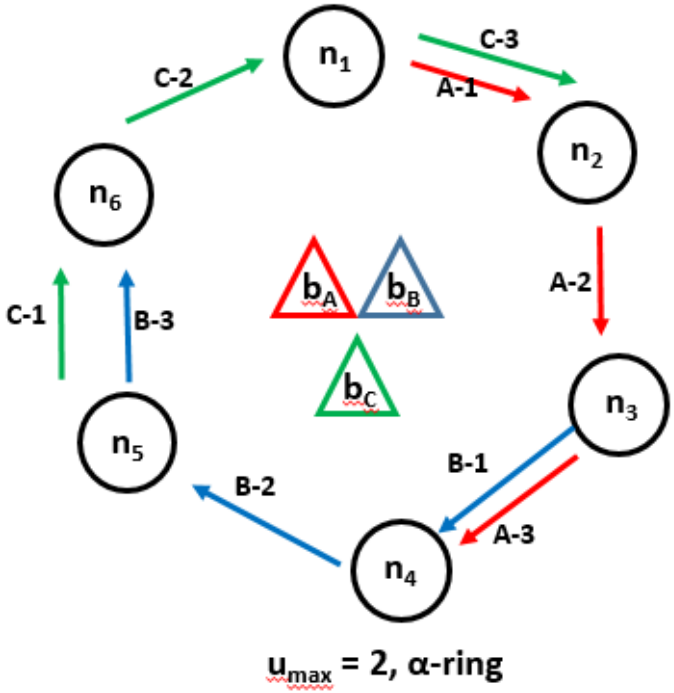
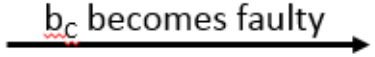


Fig. 14 Beta Ring with 6 nodes IInd time slot
K=1, 2. ***t = 2.*** **$N_\beta = 6$**
c3 = 3, 6
c4 = 4, 1
c5 = 5, 2





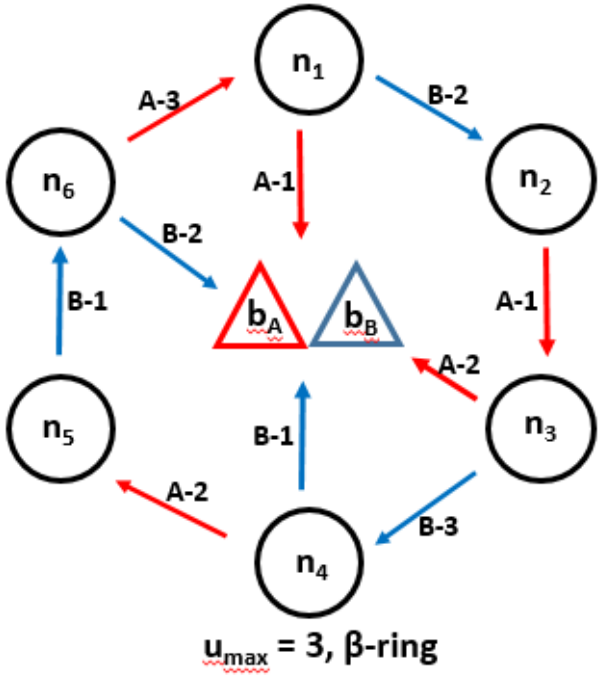


Fig. 15 Alpha Ring with 6 nodes IIIrd time slot
K=1, 2, 3. ***t = 3.*** **$N_\alpha = 6$**
$c_1 = 3, 5, 1$
$c_2 = 4, 6, 6$

Fig. 16 Beta Ring with 6 nodes IIIrd time slot
K=1, 2. ***t = 3.*** **$N_\beta = 6$**
$c3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N\beta|)) = \phi$
$c4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N\beta|)) = 6, 3$
$c5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N\beta|)) = 1, 4$

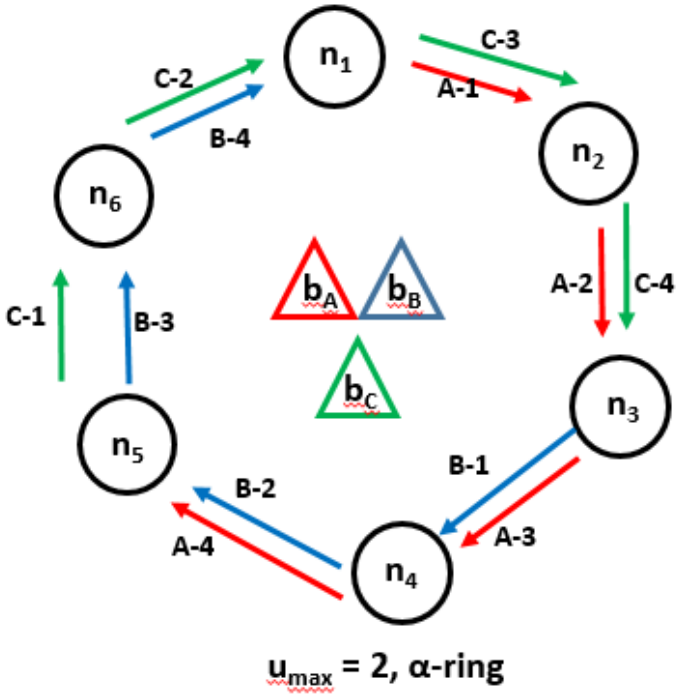


Fig. 17 Alpha Ring with 6 nodes IVth time slot
$K=1, 2, 3.$ $t = 4.$ $N_\alpha = 6$
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 4, 6, 2$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 5, 1, 3$

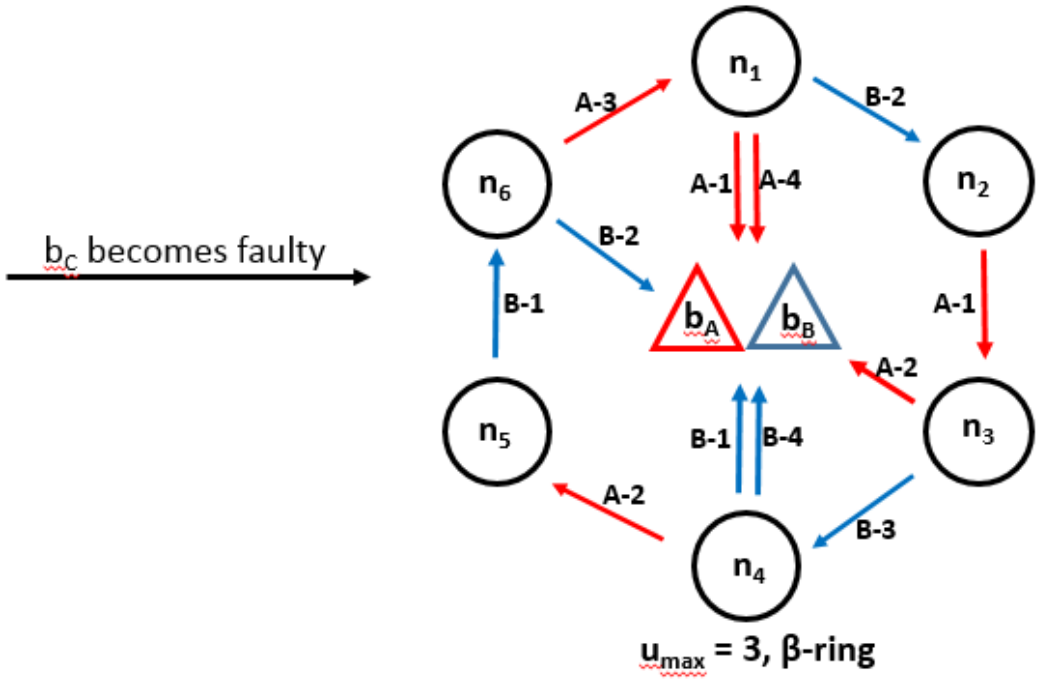


Fig. 18 Beta Ring with 6 nodes IVth time slot
$K=1, 2.$ $t = 4.$ $N_\beta = 6$
$c3 = (1 + \text{mod}(3(K\text{-}1) + 2(t\text{-}1), |N\beta|)) = 1,4$
$c4 = (1 + \text{mod}(3(K\text{-}1) + 2(t\text{-}1) + 1, |N\beta|)) = \phi$
$c5 = (1 + \text{mod}(3(K\text{-}1) + 2(t\text{-}1) + 2, |N\beta|)) = \phi$

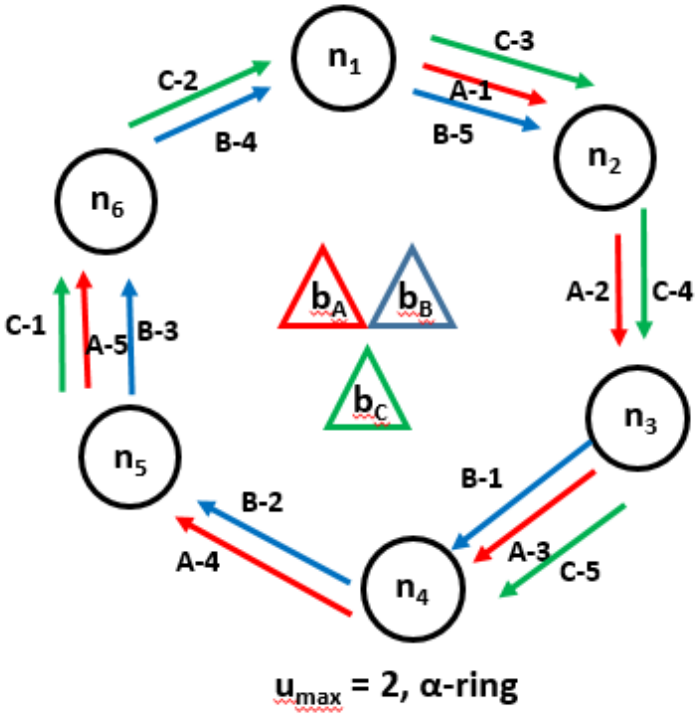


Fig. 19 Alpha Ring with 6 nodes IVth time slot
K=1, 2, 3. ***t = 5.*** **$N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 5, 1, 3$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 6, 2, 4$

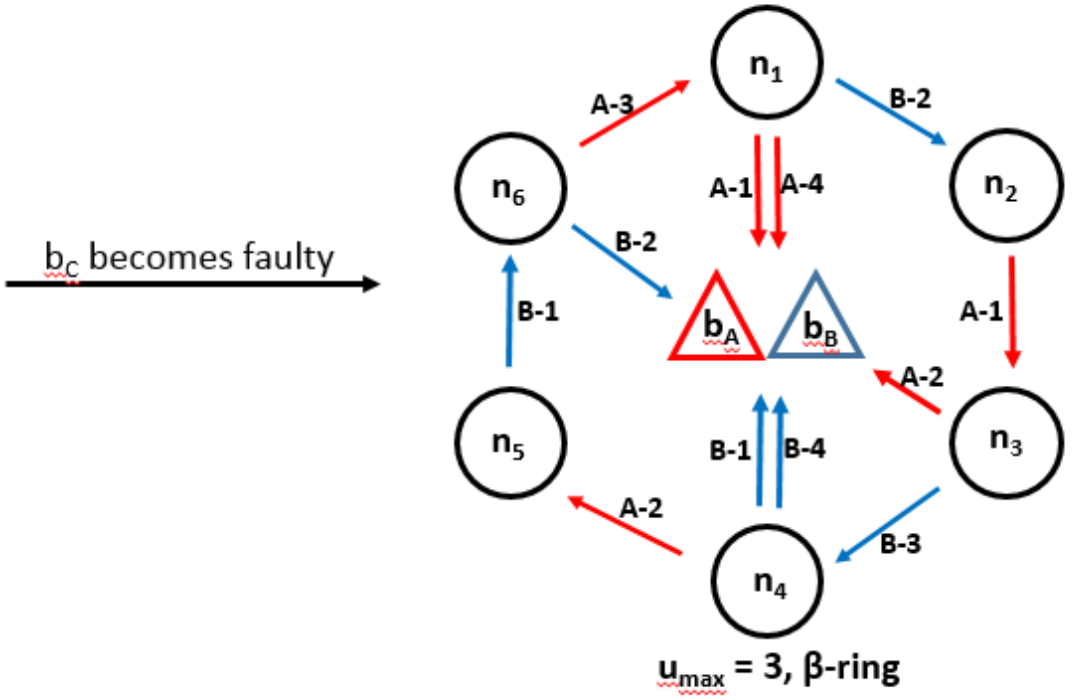
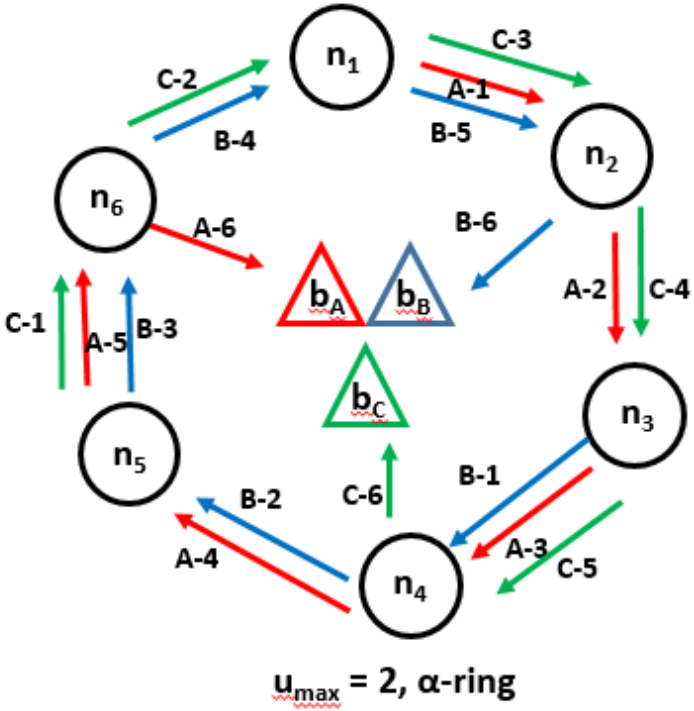
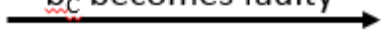


Fig. 20 Beta Ring with 6 nodes IVth time slot
K=1, 2. ***t = 4.*** **$N_\beta = 6$**
$c3 = (1 + \text{mod}(3(K\text{-}1) + 2(t\text{-}1), |N\beta|)) = \phi$
$c4 = (1 + \text{mod}(3(K\text{-}1) + 2(t\text{-}1) + 1, |N\beta|)) = \phi$
$c5 = (1 + \text{mod}(3(K\text{-}1) + 2(t\text{-}1) + 2, |N\beta|)) = \phi$





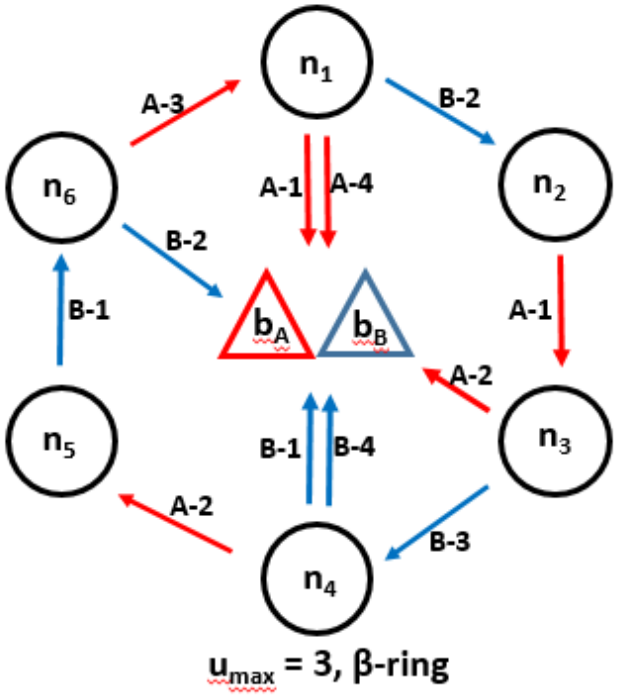


Fig. 21 Alpha Ring with 6 nodes VIth time slot
***K*=1, 2, 3. *t* = 6. $N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K-1) + t-1, |N_\alpha|)) = 6, 2, 4$
$c_2 = (1 + \text{mod}(2(K-1) + t, |N_\alpha|)) = \phi$

Fig. 22 Beta Ring with 6 nodes IVth time slot
***K*=1, 2. *t* = 4. $N_\beta = 6$**
$c3 = (1 + \text{mod}(3(K-1) + 2(t-1), |N\beta|)) = \phi$
$c4 = (1 + \text{mod}(3(K-1) + 2(t-1) + 1, |N\beta|)) = \phi$
$c5 = (1 + \text{mod}(3(K-1) + 2(t-1) + 2, |N\beta|)) = \phi$

CASE II: NODE FAULTY

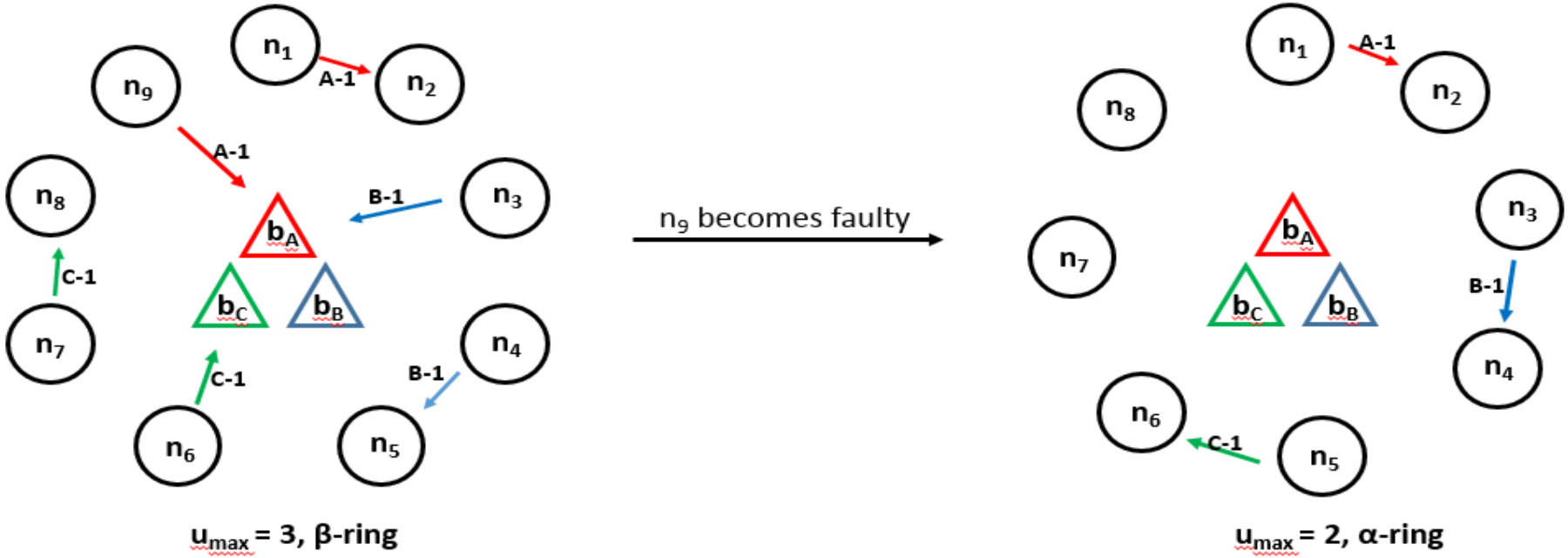
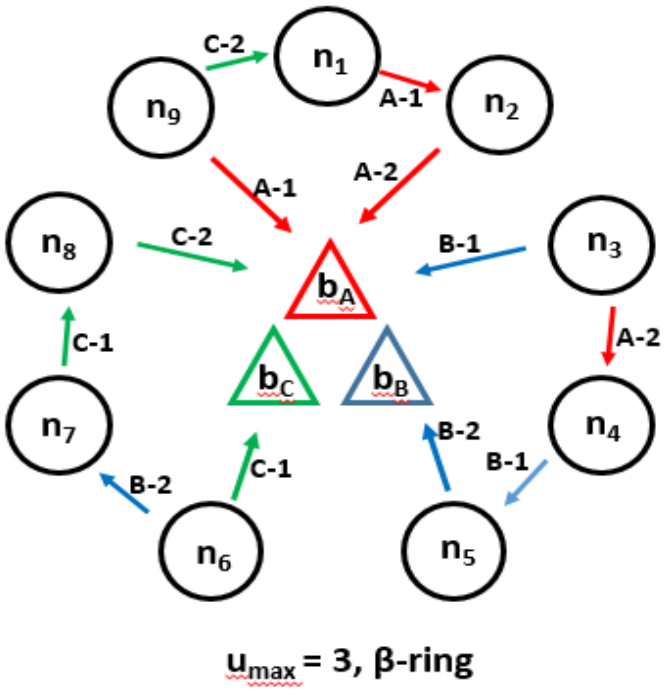
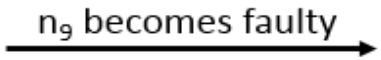


Fig. 23 Beta Ring with 9 nodes Ist time slot
K=1, 2, 3. ***t = 1.*** **$N_\beta = 9$**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =1, 4, 7
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = 2, 5, 8
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =3, 6, 9

Fig. 24 Alpha Ring with 9 nodes Ist time slot
K=1, 2, 3. ***t =1.*** **$N_\alpha = 8$**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 1,3,5
c_2 = (1 + mod(2(*K*-1) + t, |N_α|)) = 2,4, 6





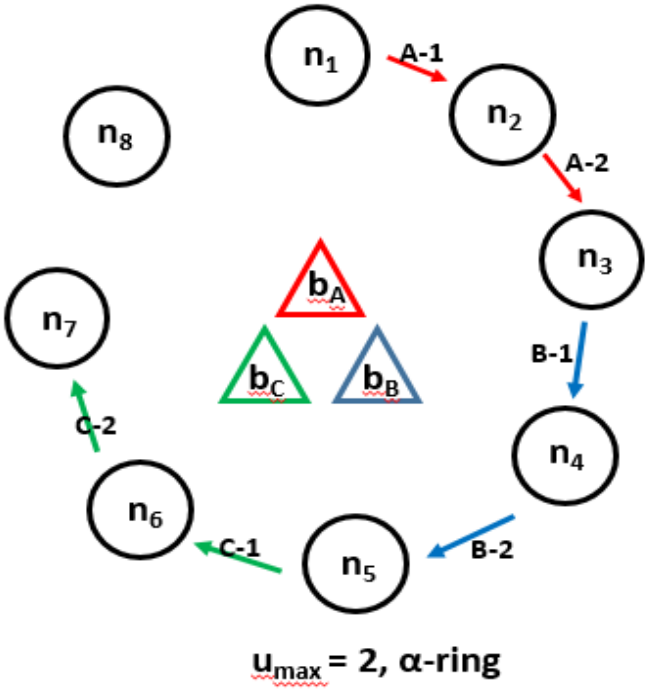
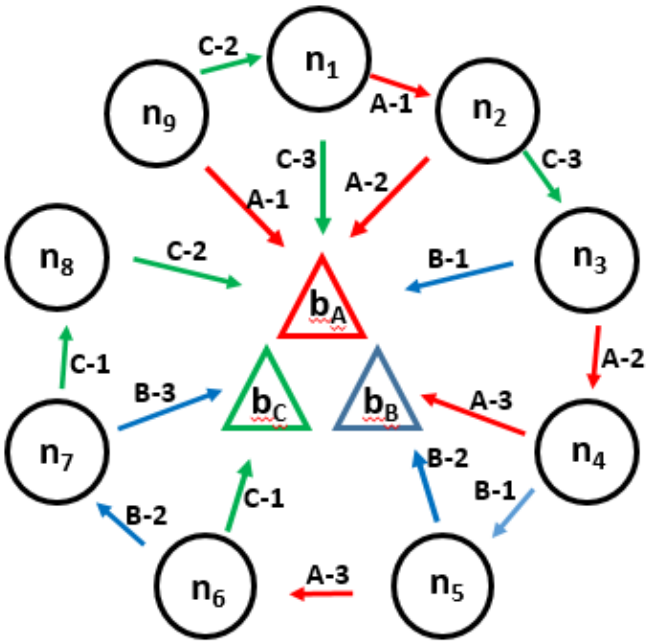


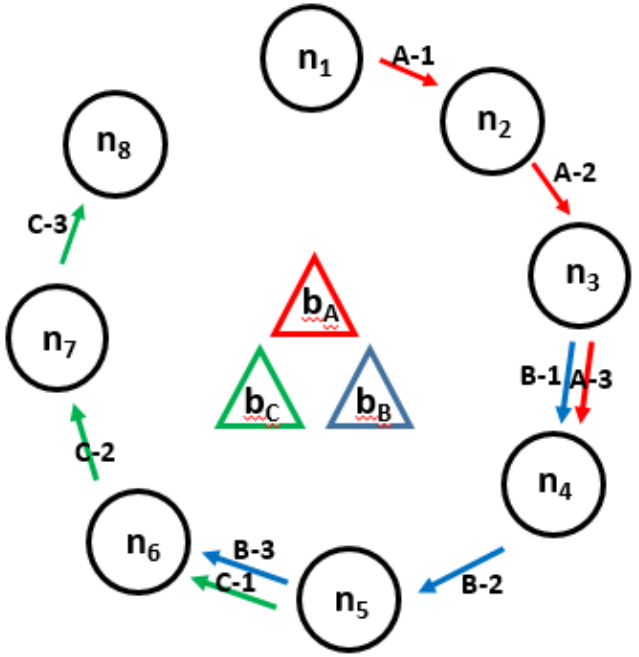
Fig. 25 Beta Ring with 9 nodes IInd time slot
K=1, 2, 3. ***t = 2.*** **$N_\beta = 9$**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =3, 6, 9
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = 4, 7, 1
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =2, 5, 8

Fig. 26 Alpha Ring with 9 nodes IInd time slot
K=1, 2, 3. ***t = 2.*** **$N_\alpha = 8$**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 2, 4, 6
c_2 = (1 + mod(2(*K-1*) + t, |N_α|)) = 3, 5, 7



u_{max} = 3, β-ring

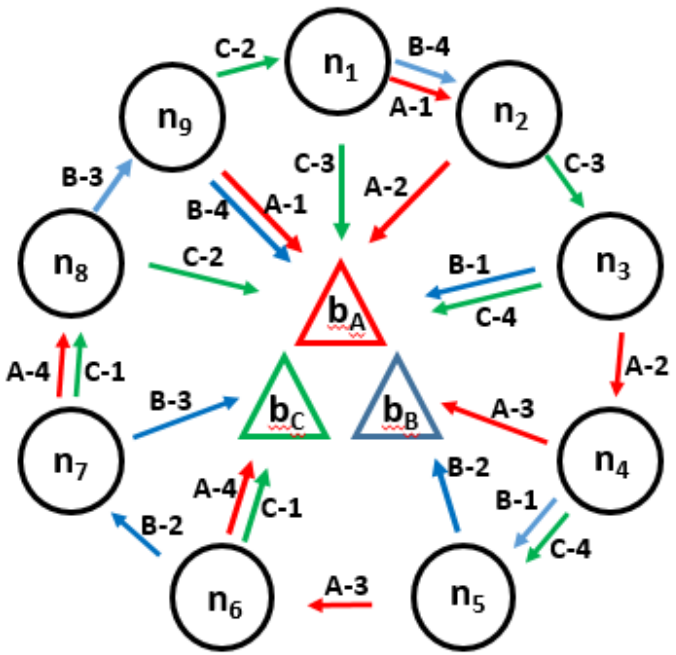
n9 becomes faulty



u_{max} = 2, α-ring

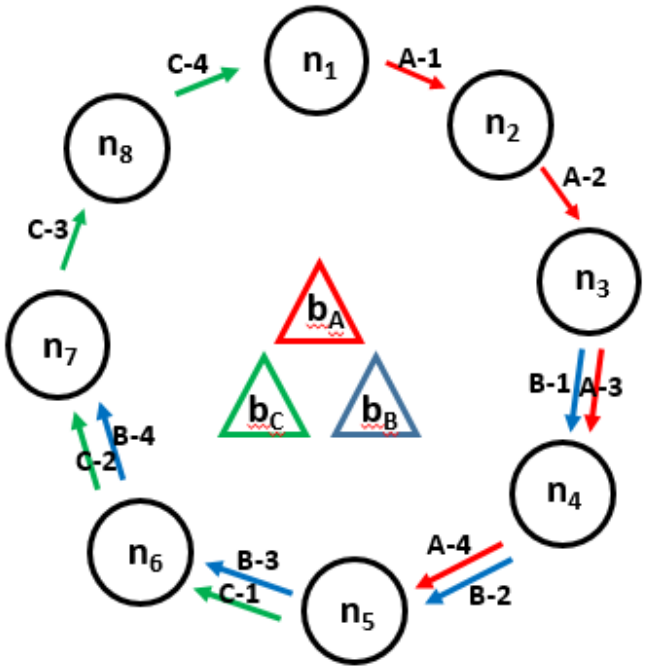
Fig. 27 Beta Ring with 9 nodes III[rd] time slot
K=1, 2, 3. ***t = 3.*** **$N_\beta = 9$**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =5, 8, 2
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = 6, 9, 3
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =4, 7, 1

Fig. 28 Alpha Ring with 8 nodes III[rd] time slot
K=1, 2, 3. ***t =3.*** **$N_\alpha = 8$**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 3,5,7
c_2 = (1 + mod(2(*K-1*) + t, |N_α|)) = 4,6, 8



u_{max} = 3, β-ring

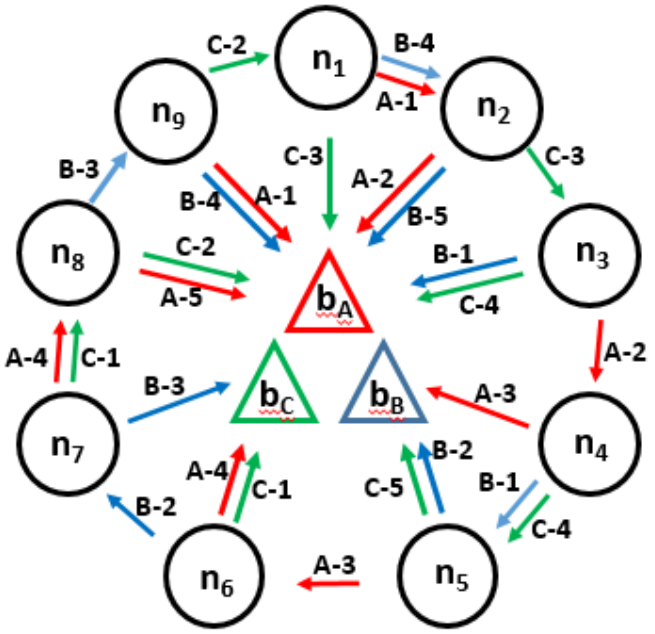
n9 becomes faulty

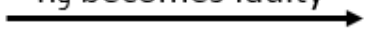


u_{max} = 2, α-ring

Fig. 29 Beta Ring with 9 nodes IVth time slot
K=1, 2, 3. ***t = 4.*** **N_β = 9**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =7, 1, 4
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = 8, 2, 5
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =6, 9, 3

Fig. 30 Alpha Ring with 8 nodes IVth time slot
K=1, 2, 3. ***t =4.*** **N_α = 8**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 4,6,8
c_2 = (1 + mod(2(*K-1*) + t, |N_α|)) = 5,7, 1





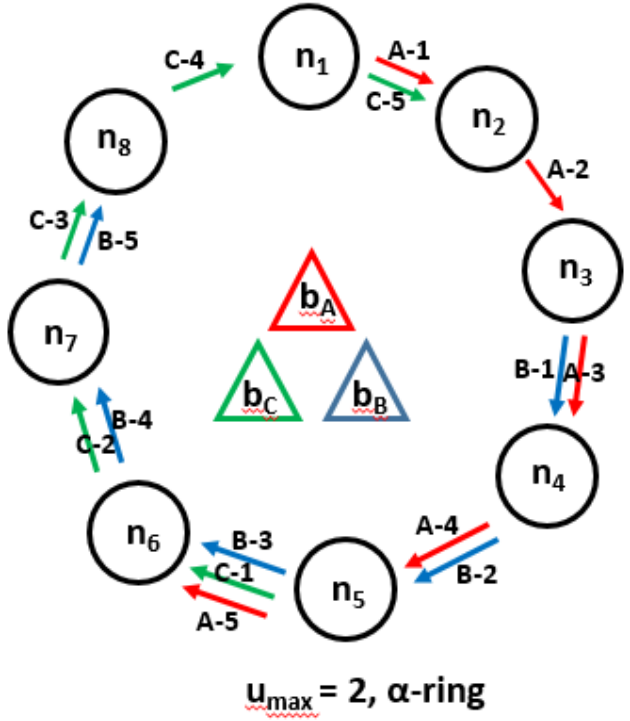


Fig. 31 Beta Ring with 9 nodes Vth time slot
K=1, 2, 3. ***t = 5.*** **$N_\beta = 9$**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =ϕ
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = ϕ
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =8, 2, 5

Fig. 32 Alpha Ring with 8 nodes Vth time slot
K=1, 2, 3. ***t =5.*** **$N_\alpha = 8$**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 5, 7, 1
c_2 = (1 + mod(2(*K-1*) + t, |N_α|)) = 6, 8, 2

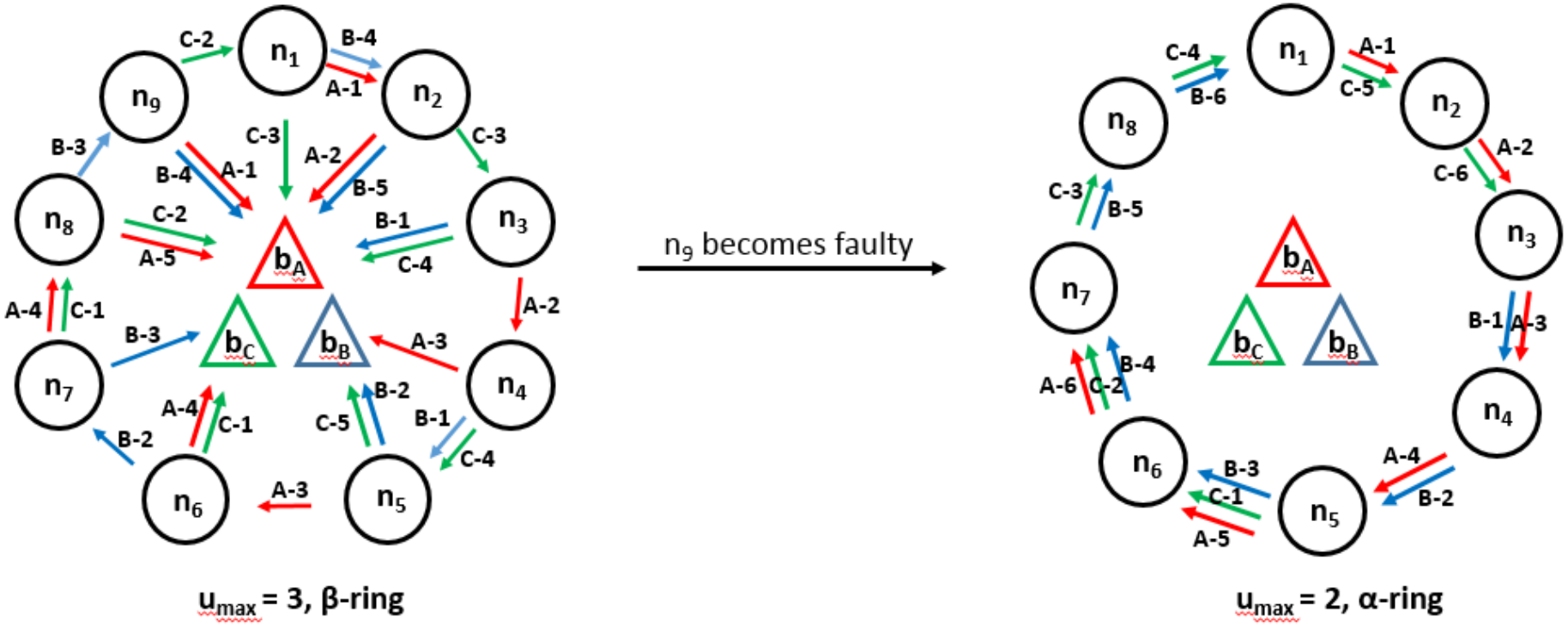
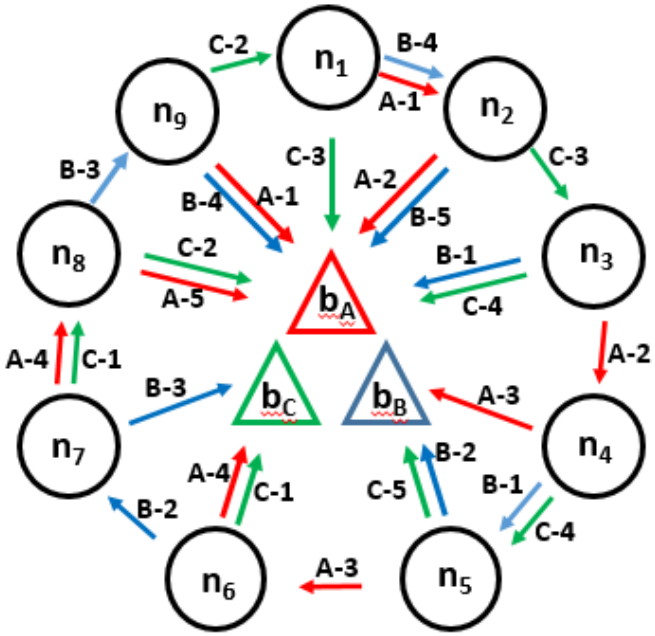


Fig. 33 Beta Ring with 9 nodes Vth time slot
K=1, 2, 3. ***t = 5.*** **$N_\beta = 9$**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =ϕ
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = ϕ
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =ϕ

Fig. 34 Alpha Ring with 8 nodes VIth time slot
K=1, 2, 3. ***t =6.*** **$N_\alpha = 8$**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 6, 8, 2
c_2 = (1 + mod(2(*K*-1) + t, |N_α|)) = 7, 1, 3



n_9 becomes faulty

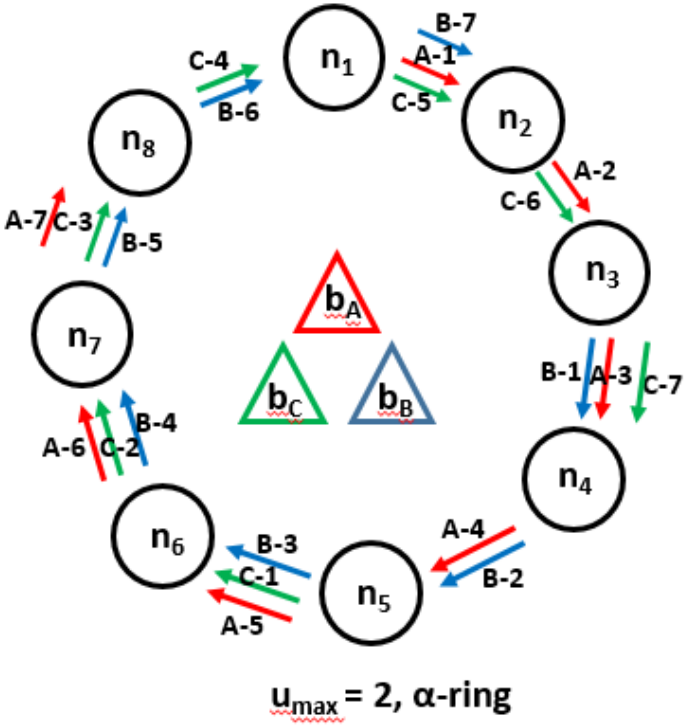
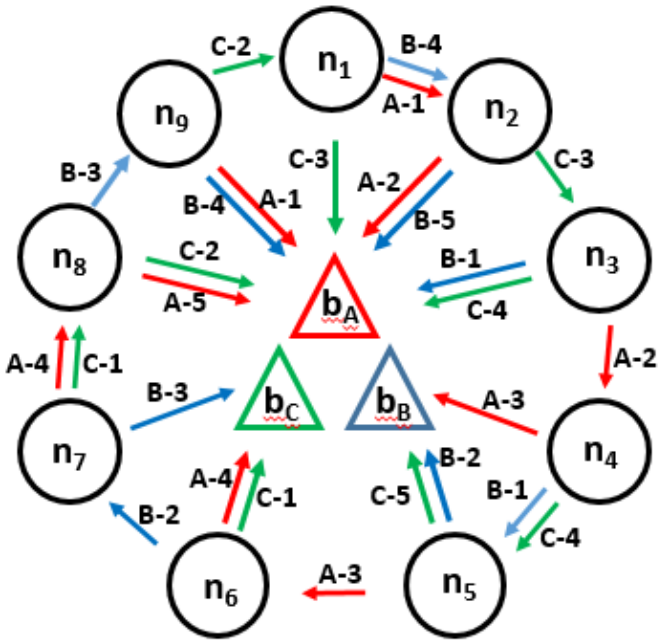


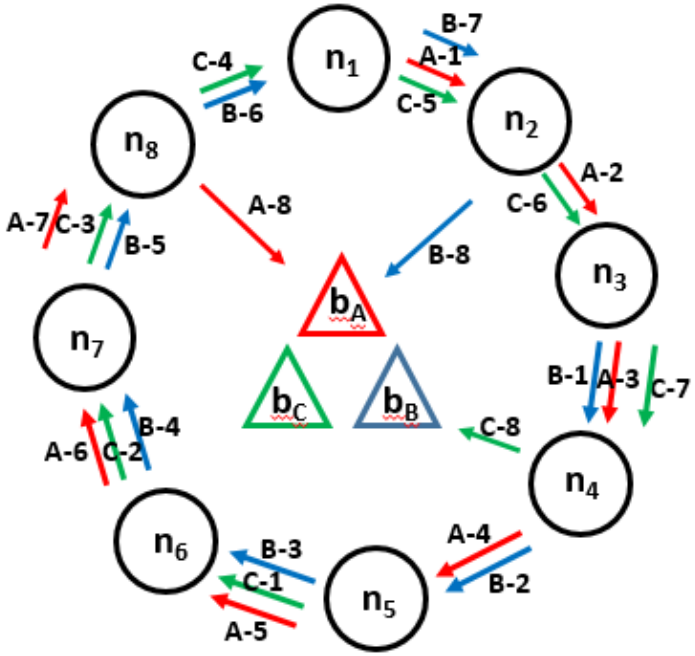
Fig. 23 Beta Ring with 9 nodes Vth time slot
K=1, 2, 3. ***t =5.*** **$N_\beta = 9$**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =ϕ
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = ϕ
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =ϕ

Fig. 36 Alpha Ring with 6 nodes VIIth time slot
K=1, 2, 3. ***t =7.*** **$N_\alpha = 8$**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 7, 1, 3
c_2 = (1 + mod(2(*K*-1) + t, |N_α|)) = 8, 2, 4



u_{max} = 3, β-ring

n9 becomes faulty

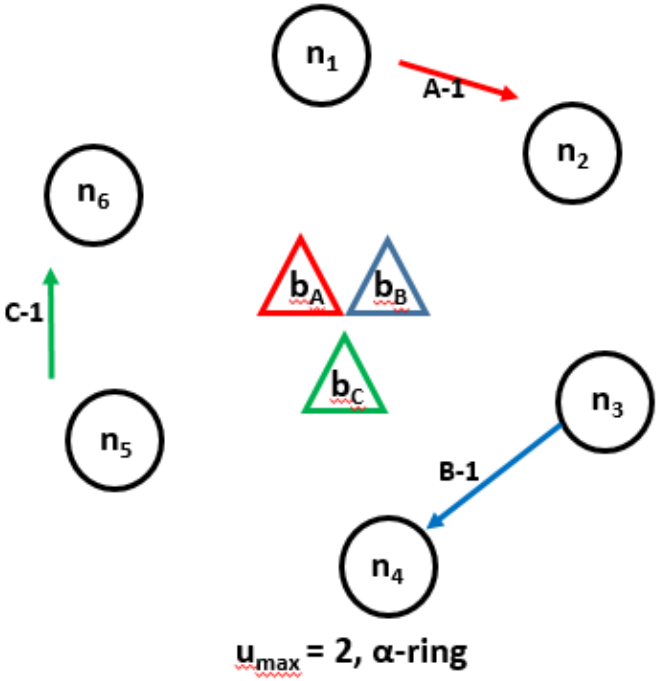


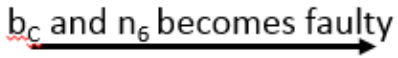
u_{max} = 2, α-ring

Fig. 37 Beta Ring with 9 nodes Vth time slot
***K*=1, 2, 3. *t* = 5. $N_\beta = 9$**
c3 = (1 + mod(3(K-1) + 2(t-1), |Nβ|)) =ϕ
c4 = (1 + mod(3(K-1) + 2(t-1) + 1, |Nβ|)) = ϕ
c5 = (1 + mod(3(K-1) + 2(t-1) + 2, |Nβ|)) =ϕ

Fig. 38 Alpha Ring with 6 nodes VIIIth time slot
***K*=1, 2, 3. *t* =8. $N_\alpha = 8$**
c_1 = (1 + mod(2(*K*-1) + t-1, |N_α|)) = 2, 4, 8
c_2 = (1 + mod(2(*K*-1) + t, |N_α|)) = ϕ

CASE 3: NODE AND BASE STATION FAULTY





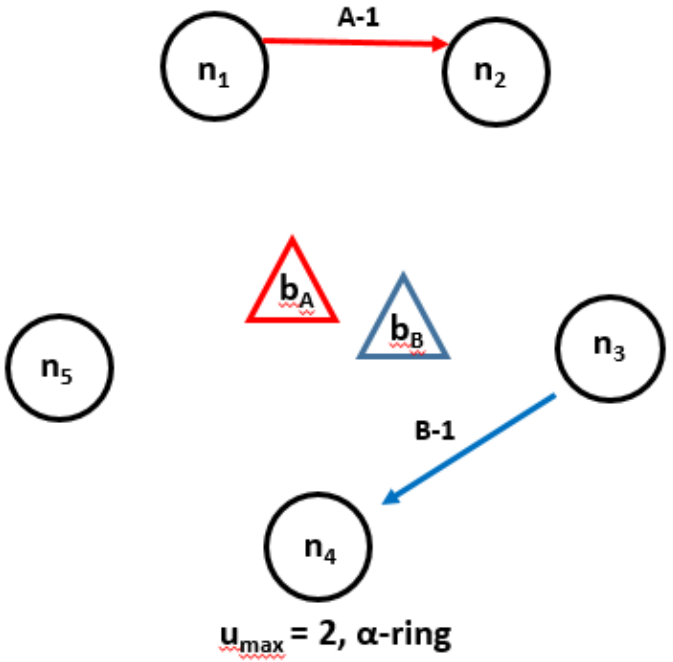
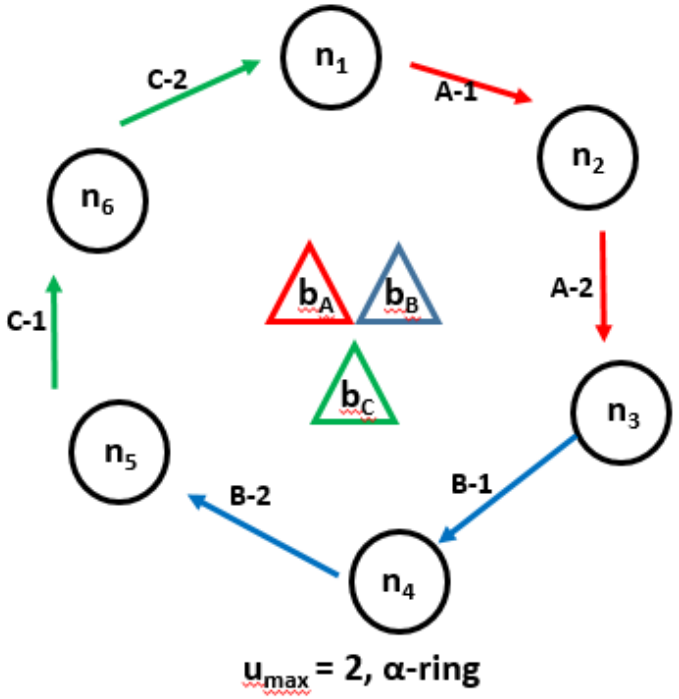
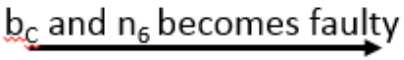


Fig. 39 Alpha Ring with 6 nodes Ist time slot
K=1, 2, 3. ***t =1.*** **$N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 1, 3, 5$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 2, 4, 6$

Fig. 40 Alpha Ring with 5 nodes Ist time slot
K=1, 2. ***t =1.*** **$N_\alpha = 5$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 7, 1, 3$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 8, 2, 4$





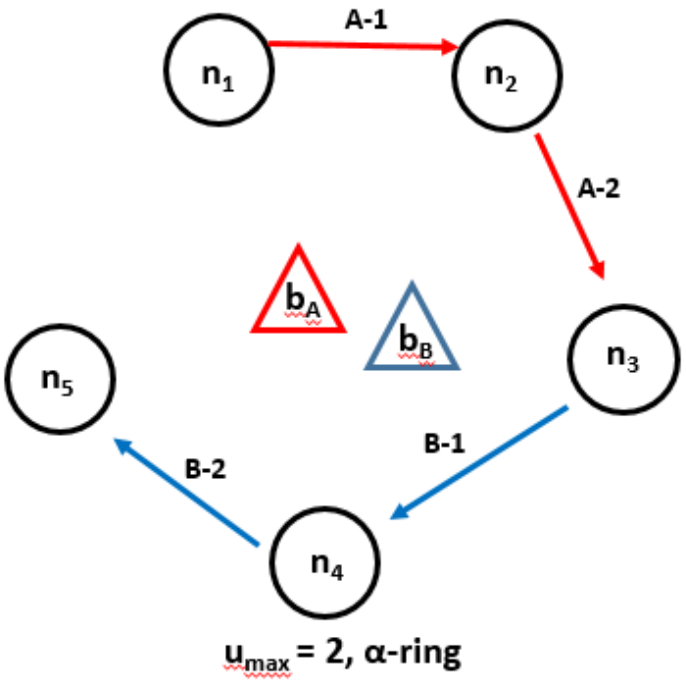
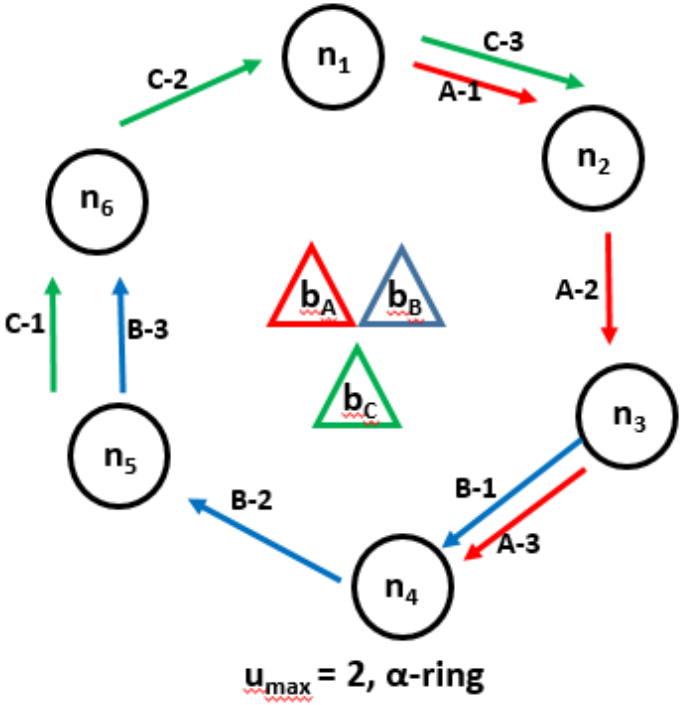
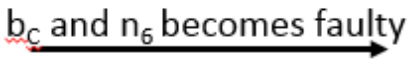


Fig. 41 Alpha Ring with 6 nodes IInd time slot
K=1, 2, 3. ***t =2.*** **$N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 2, 4, 6$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 3, 5, 1$

Fig. 42 Alpha Ring with 5 nodes IInd time slot
K=1, 2. ***t =2.*** **$N_\alpha = 5$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 2, 4$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 3, 5$





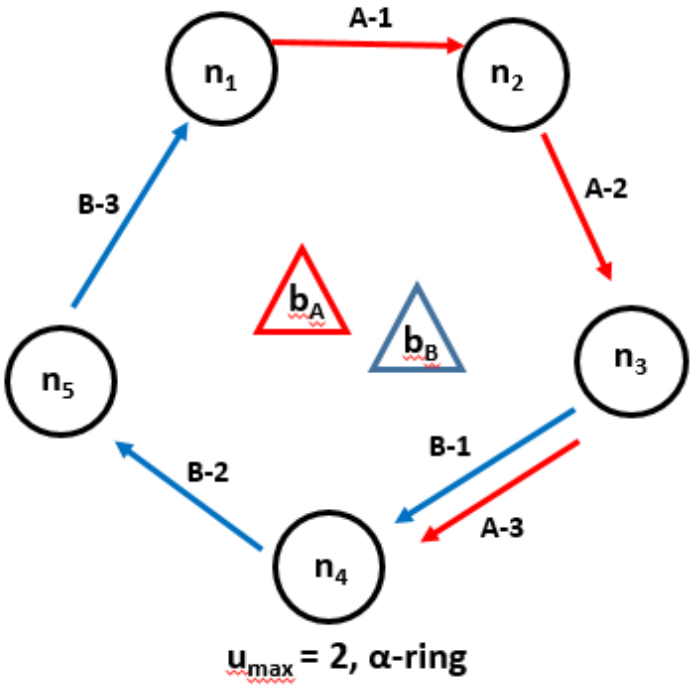
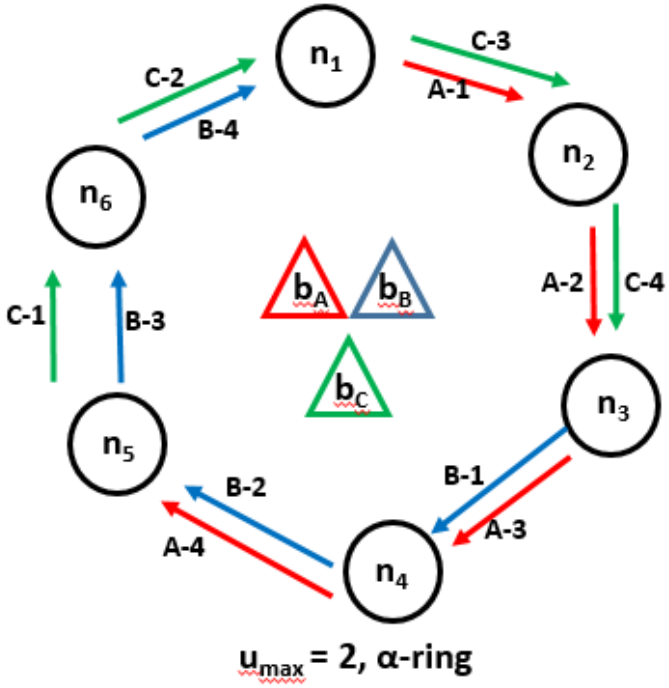


Fig. 43 Alpha Ring with 6 nodes III[rd] time slot
K=1, 2, 3. ***t =3.*** **$N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 3, 5, 1$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 4, 6, 2$

Fig. 44 Alpha Ring with 5 nodes III[rd] time slot
K=1, 2. ***t =3.*** **$N_\alpha = 5$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 3, 5$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 4, 1$



bC and n6 becomes faulty

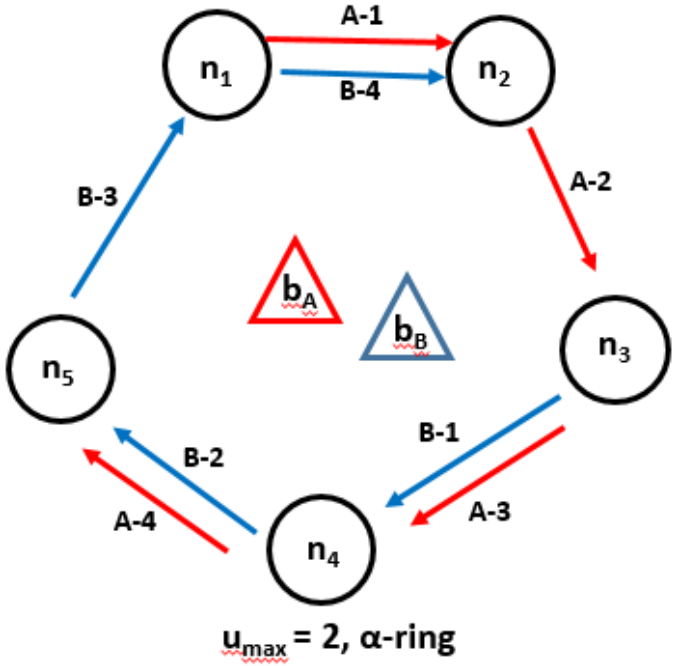
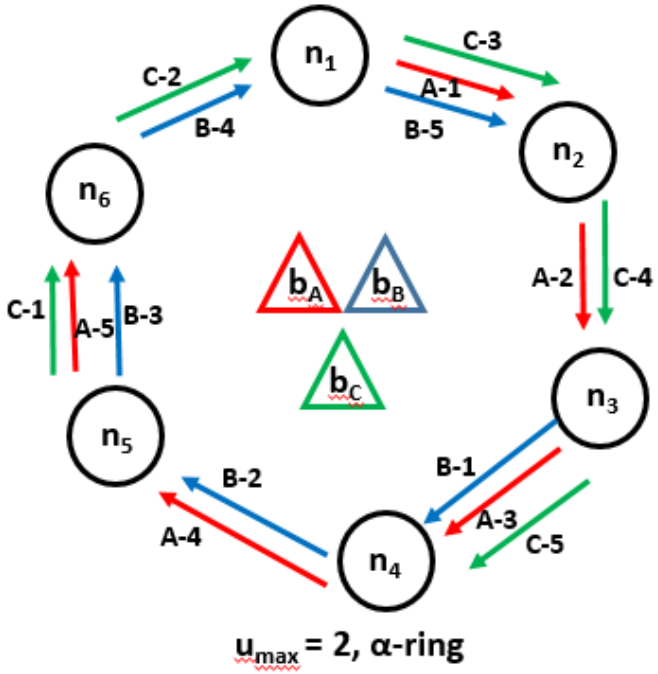
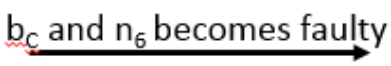


Fig. 45 Alpha Ring with 6 nodes IVth time slot
K=1, 2, 3. ***t =4.*** **$N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 4, 6, 2$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 5, 1, 3$

Fig. 46 Alpha Ring with 5 nodes IVth time slot
K=1, 2. ***t =4.*** **$N_\alpha = 5$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 4, 1$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 5, 2$





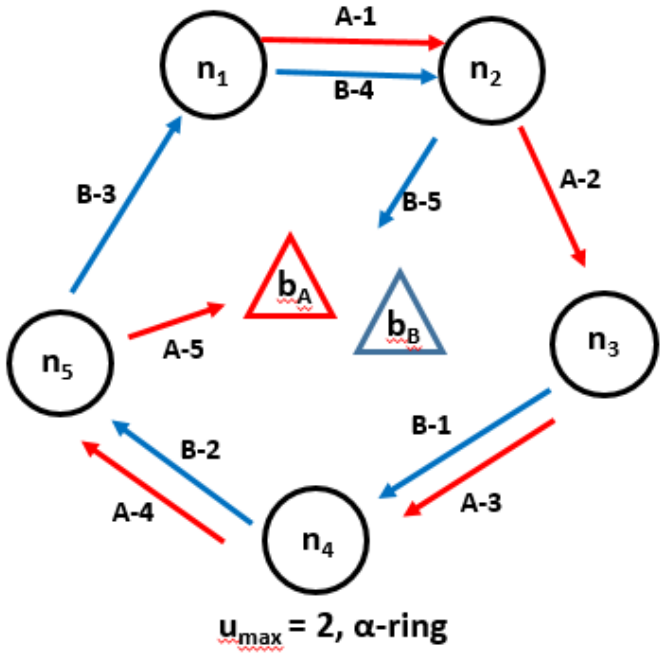
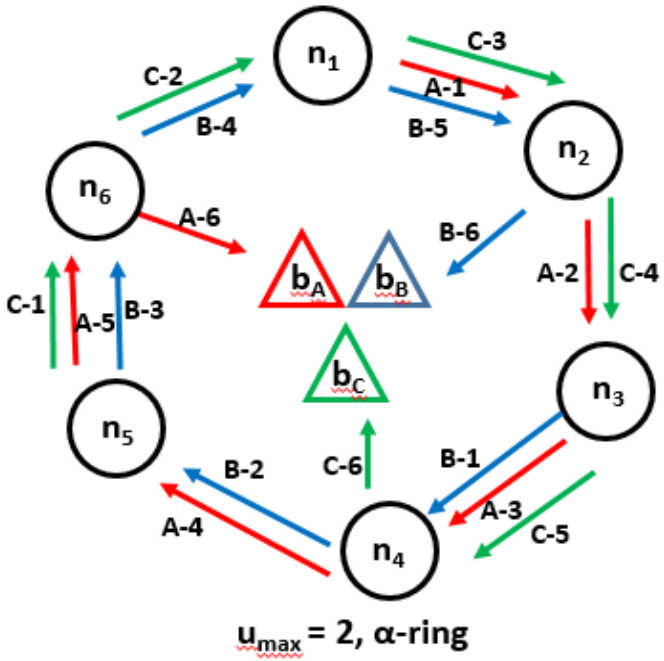


Fig. 47 Alpha Ring with 6 nodes Vth time slot
K=1, 2, 3. ***t =5.*** **$N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 5, 1, 3$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = 6, 2, 4$

Fig. 48 Alpha Ring with 5 nodes Vth time slot
K=1, 2. ***t =5.*** **$N_\alpha = 5$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 2,5$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = \phi$



bC and n6 becomes faulty

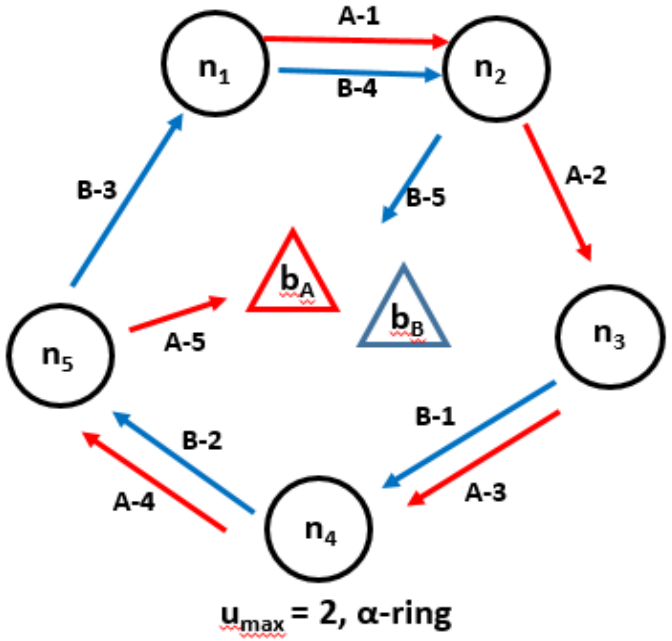


Fig. 49 Alpha Ring with 6 nodes VIth time slot
K=1, 2, 3. ***t =6.*** **$N_\alpha = 6$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 2, 4, 6$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = \phi$

Fig. 50 Alpha Ring with 5 nodes Vth time slot
K=1, 2. ***t =5.*** **$N_\alpha = 5$**
$c_1 = (1 + \text{mod}(2(K\text{-}1) + t\text{-}1, |N_\alpha|)) = 2, 5$
$c_2 = (1 + \text{mod}(2(K\text{-}1) + t, |N_\alpha|)) = \phi$

TEST DATA SET USED & RESULTS

For each of the simulations performed below for all three cases we have considered initial total number of nodes equal to 300, total number of data streams equal to 10 and time slots are generated using (4) & (5). For the first case where base station is faulty, nodes are kept at 300 while data streams are decremented successively in each iteration by 2 and suitable values of time slots are calculated. For the second case where nodes are faulty, base stations or data streams are kept constant at 10 while nodes are decremented successively in each iteration by 2. For the third case where both nodes and base stations are faulty a nested concept is used to first make a single base station faulty and decrementing number of nodes successively by 1 and in the outermost iteration increasing faulty base stations by a factor of 1 successively.

From the following set of simulations we have observed how the 3D graph structure changes with subject to changes made in number of nodes and base stations. As for the first case we have observed that the graph structure changes drastically as the number of base stations or data streams vary which gives the indication that it has significant effect on the performance of the network as the number of time slots decreases. Furthermore, in the case where nodes become faulty there is almost slight or no difference in the 3D graph structure. The only place where variation is observed is on the top left edges of the graph near the 10^{th} base station and the values near 300^{th} node. Only a slight difference in the values of time-slots is observed. In the third case where both become faulty, for figures C(i)-C(v) the graph structure changes slightly between the middle and upper ends with only a slight difference in the values of time slots. In figures C(vi)-C(x) where two base station are faulty a significant difference is observed on the top right corner of the graph with only slight changes in the value of time slots. This variation increases in the manner as from top right to top left and then downward. With this our observation is that as the number of nodes changes it has lesser impact on the number of time slots and thus on the performance, but a change in the number of base stations or data streams decreases time slots significantly. Also when two cases are combined the number of time slots decrease with decrease in the number of data streams.

RESULTS: CASE A: BASE STATION FAULTY

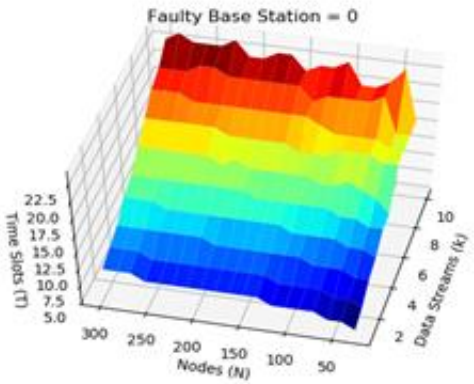


Fig. A(i)
Number of nodes (N) = 300
Faulty Base Station ($\widehat{b}$) = 0
Total Data Streams (k) = 10

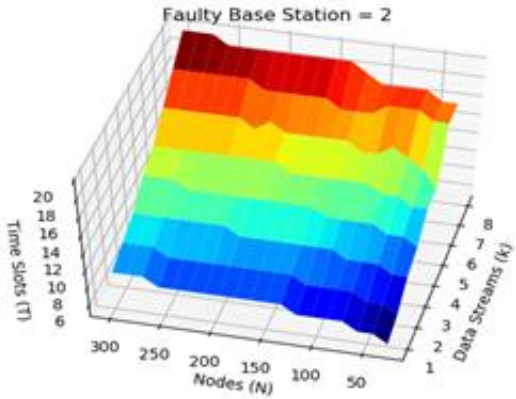


Fig. A(ii)
Number of nodes (N) = 300
Faulty Base Station ($\widehat{b}$) = 2
Total Data Streams (k) = 8

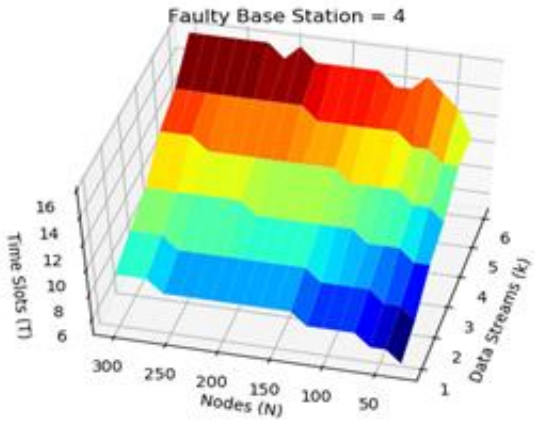


Fig. A(iii)
Number of nodes (N) = 300
Faulty Base Station ($\widehat{b}$) = 4
Total Data Streams (k) = 6

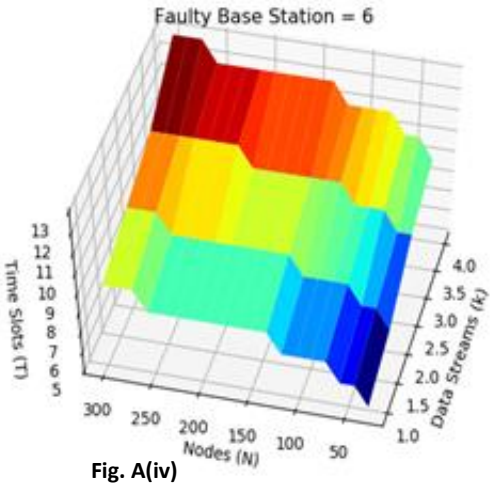


Fig. A(iv)
Number of nodes (N) = 300
Faulty Base Station ($\widehat{b}$) = 6
Total Data Streams (k) = 4

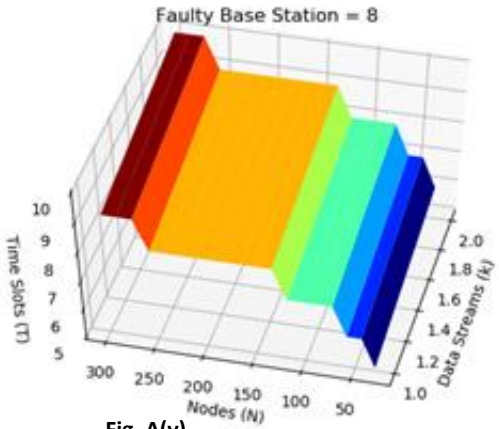


Fig. A(v)
Number of nodes (N) = 300
Faulty Base Station ($\widehat{b}$) = 8
Total Data Streams (k) = 2

RESULTS: CASE B: NODE FAULTY

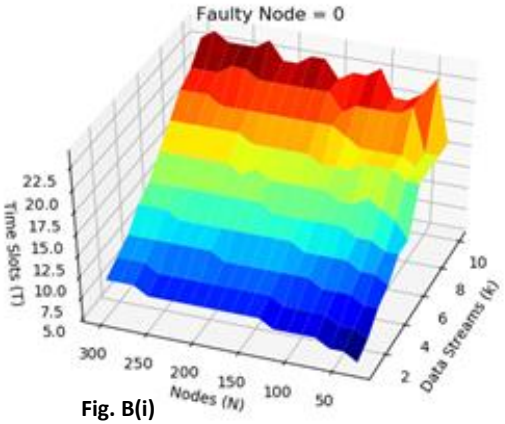


Fig. B(i)
Number of nodes (N) = 300
Faulty nodes ($\hat{n}$) = 0
Total Data Streams (k) = 10

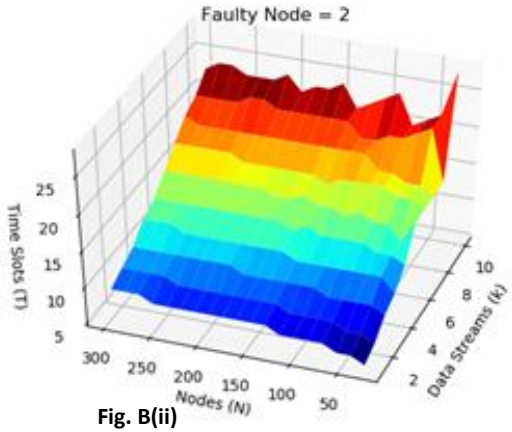


Fig. B(ii)
Number of nodes (N) = 298
Faulty nodes ($\hat{n}$) = 2
Total Data Streams (k) = 10

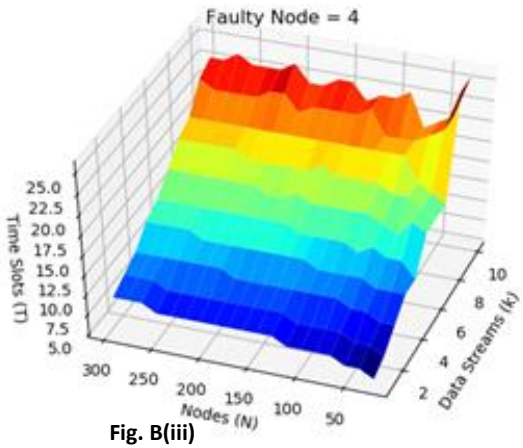


Fig. B(iii)
Number of nodes (N) = 296
Faulty nodes ($\hat{n}$) = 4
Total Data Streams (k) = 10

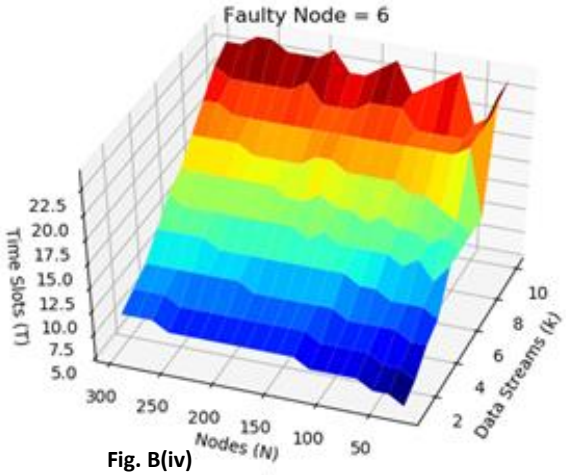


Fig. B(iv)
Number of nodes (N) = 294
Faulty nodes ($\hat{n}$) = 6
Total Data Streams (k) = 10

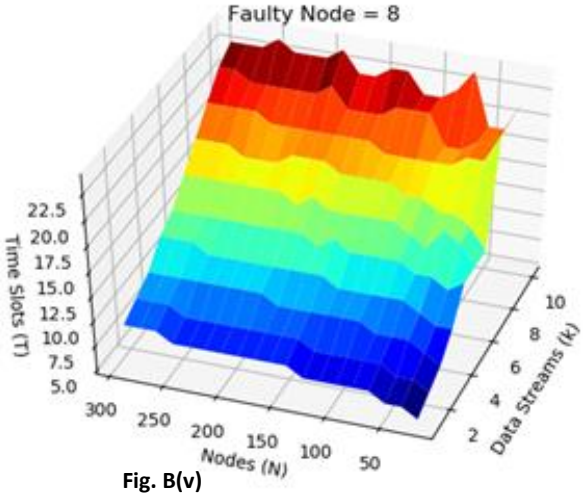


Fig. B(v)
Number of nodes (N) = 292
Faulty nodes ($\hat{n}$) = 8
Total Data Streams (k) = 10

RESULTS: CASE C: NODES FAULTY & ONE BS FAULTY

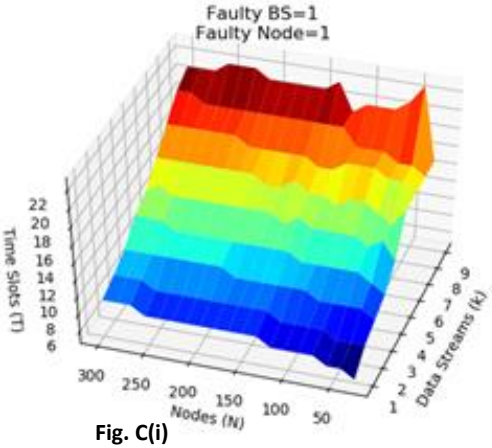


Fig. C(i)
Number of nodes (N) = 299
Faulty nodes ($\hat{n}$) = 1
Faulty base station ($\hat{b}$) = 1
Total Data Streams (k) = 9

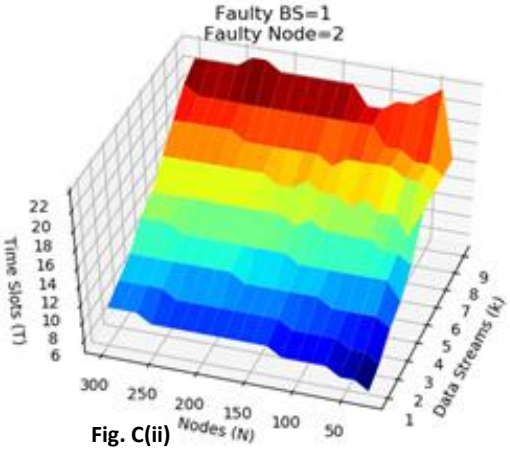


Fig. C(ii)
Number of nodes (N) = 298
Faulty nodes ($\hat{n}$) = 2
Faulty base station ($\hat{b}$) = 1
Total Data Streams (k) = 9

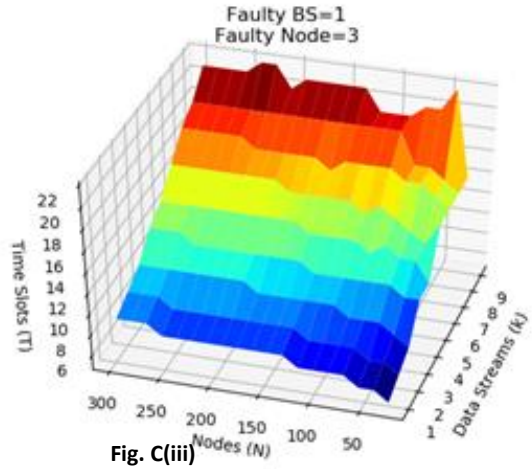


Fig. C(iii)
Number of nodes (N) = 297
Faulty nodes ($\hat{n}$) = 3
Faulty base station ($\hat{b}$) = 1
Total Data Streams (k) = 9

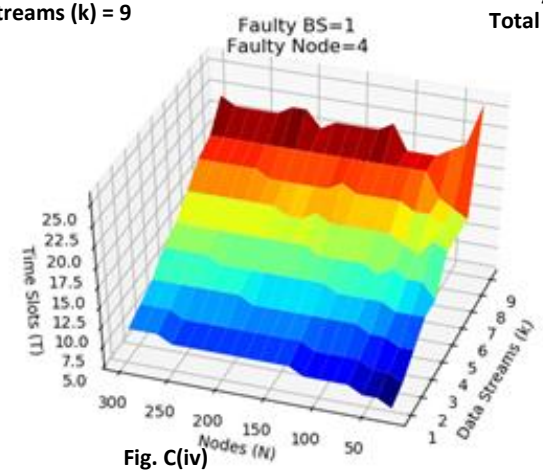


Fig. C(iv)
Number of nodes (N) = 296
Faulty nodes ($\hat{n}$) = 4
Faulty base station ($\hat{b}$) = 1
Total Data Streams (k) = 9

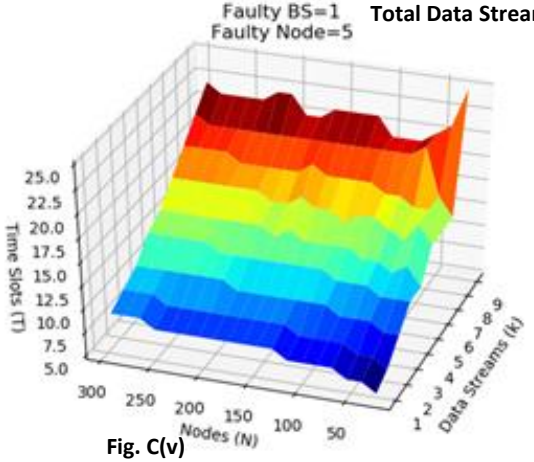


Fig. C(v)
Number of nodes (N) = 295
Faulty nodes ($\hat{n}$) = 5
Faulty base station ($\hat{b}$) = 1
Total Data Streams (k) = 9

RESULTS— NODES FAULTY & TWO BS FAULTY

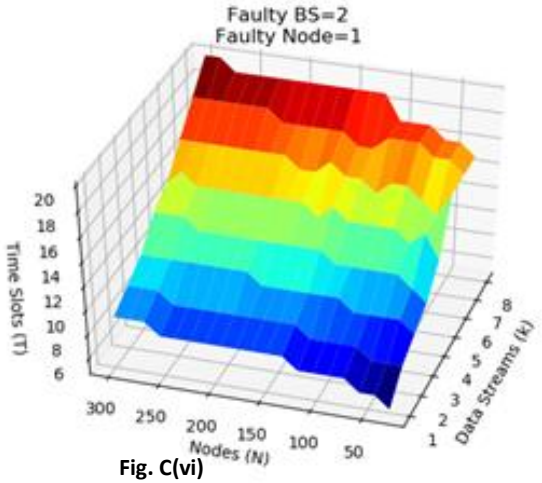


Fig. C(vi)
Number of nodes (N) = 299
Faulty nodes ($\hat{n}$) = 1
Faulty base station ($\hat{b}$) = 2
Total Data Streams (k) = 8

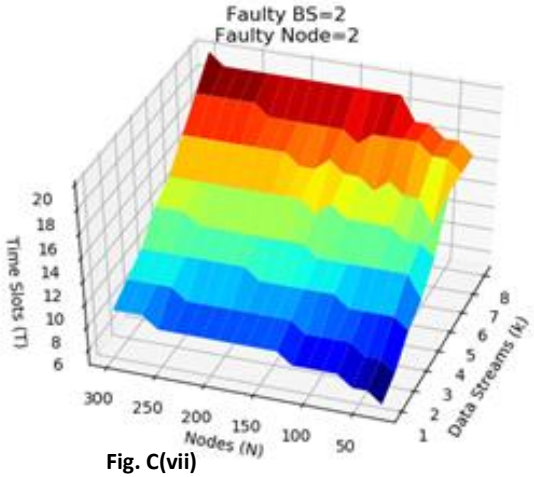


Fig. C(vii)
Number of nodes (N) = 298
Faulty nodes ($\hat{n}$) = 2
Faulty base station ($\hat{b}$) = 2
Total Data Streams (k) = 8

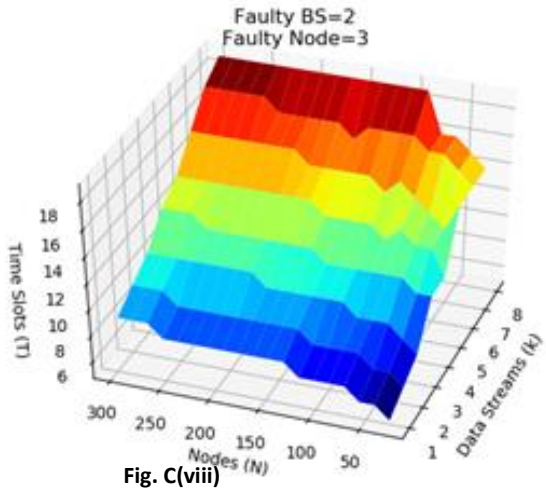


Fig. C(viii)
Number of nodes (N) = 297
Faulty nodes ($\hat{n}$) = 3
Faulty base station ($\hat{b}$) = 2
Total Data Streams (k) = 8

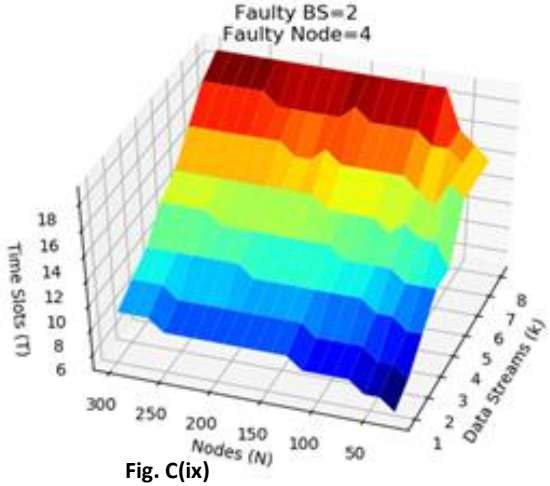


Fig. C(ix)
Number of nodes (N) = 296
Faulty nodes ($\hat{n}$) = 4
Faulty base station ($\hat{b}$) = 2
Total Data Streams (k) = 8

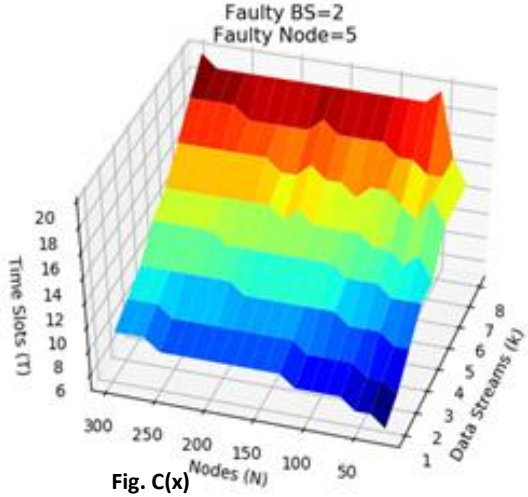


Fig. C(x)
Number of nodes (N) = 295
Faulty nodes ($\hat{n}$) = 5
Faulty base station ($\hat{b}$) = 2
Total Data Streams (k) = 8

RESULTS— NODES FAULTY & THREE BS FAULTY

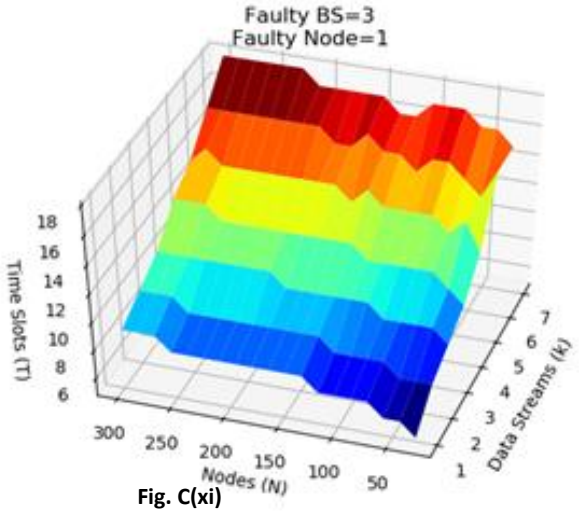


Fig. C(xi)
Number of nodes (N) = 299
Faulty nodes ($\hat{n}$) = 1
Faulty base station ($\hat{b}$) = 3
Total Data Streams (k) = 7

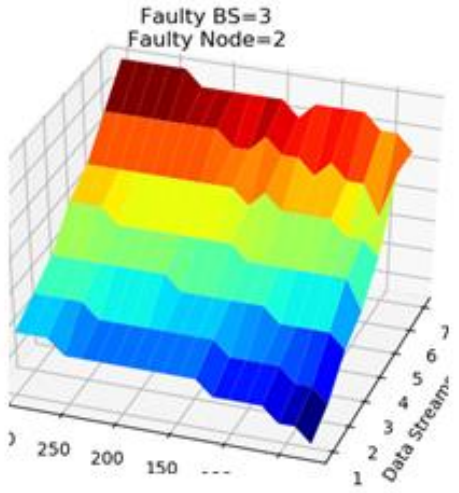


Fig. C(xii)
Number of nodes (N) = 298
Faulty nodes ($\hat{n}$) = 2
Faulty base station ($\hat{b}$) = 3
Total Data Streams (k) = 7

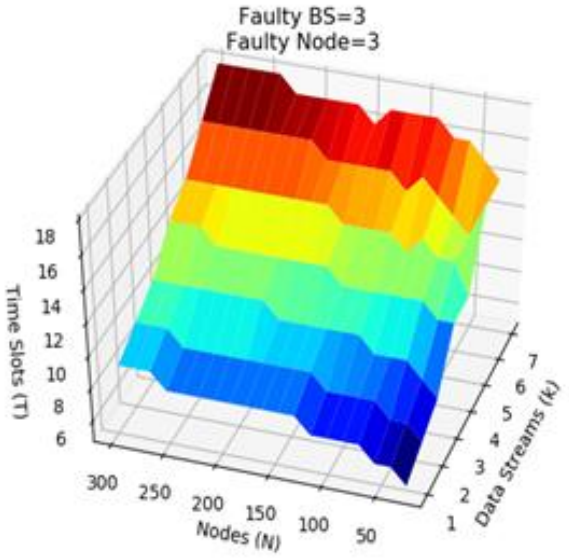


Fig. C(xiii)
Number of nodes (N) = 297
Faulty nodes ($\hat{n}$) = 3
Faulty base station ($\hat{b}$) = 3
Total Data Streams (k) = 7

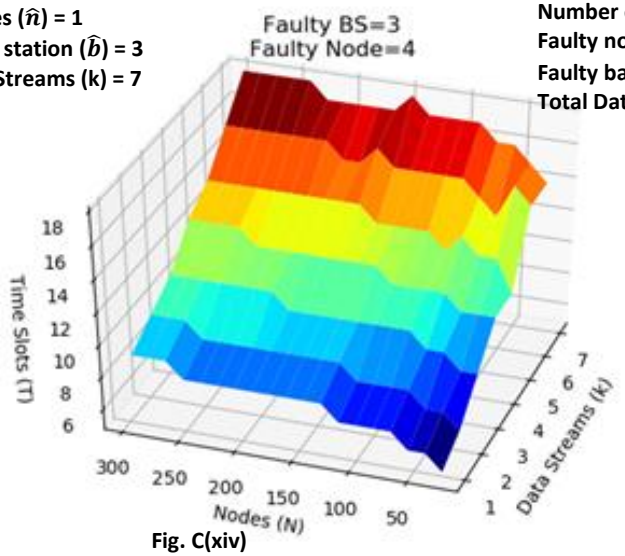


Fig. C(xiv)
Number of nodes (N) = 296
Faulty nodes ($\hat{n}$) = 4
Faulty base station ($\hat{b}$) = 3
Total Data Streams (k) = 7

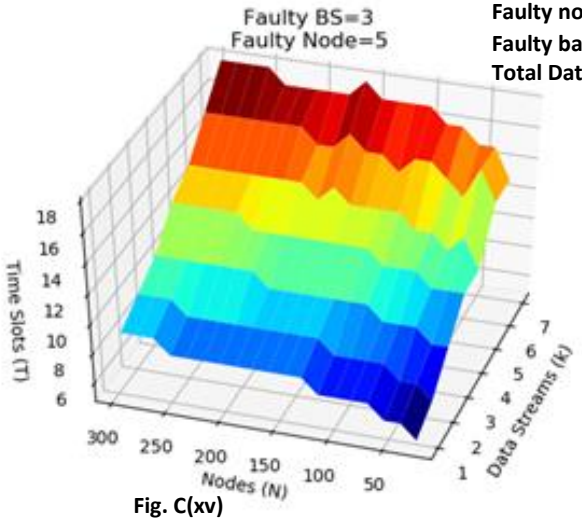


Fig. C(xv)
Number of nodes (N) = 295
Faulty nodes ($\hat{n}$) = 5
Faulty base station ($\hat{b}$) = 3
Total Data Streams (k) = 7

RESULTS— NODES FAULTY & FOUR BS FAULTY

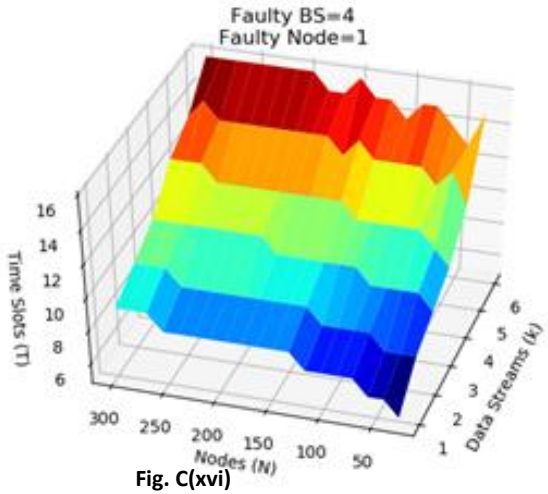


Fig. C(xvi)
Number of nodes (N) = 299
Faulty nodes ($\hat{n}$) = 1
Faulty base station ($\hat{b}$) = 4
Total Data Streams (k) = 6

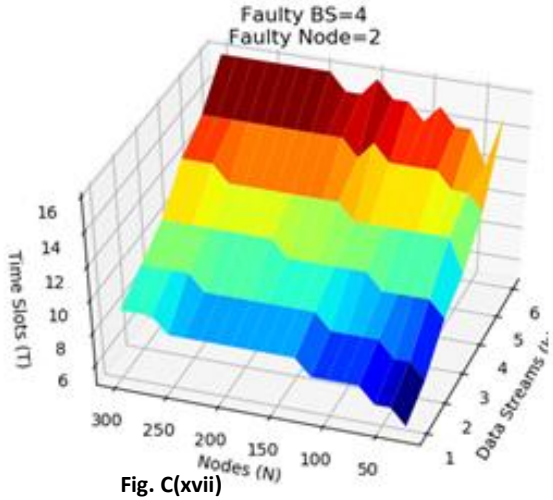


Fig. C(xvii)
Number of nodes (N) = 298
Faulty nodes ($\hat{n}$) = 2
Faulty base station ($\hat{b}$) = 4
Total Data Streams (k) = 6

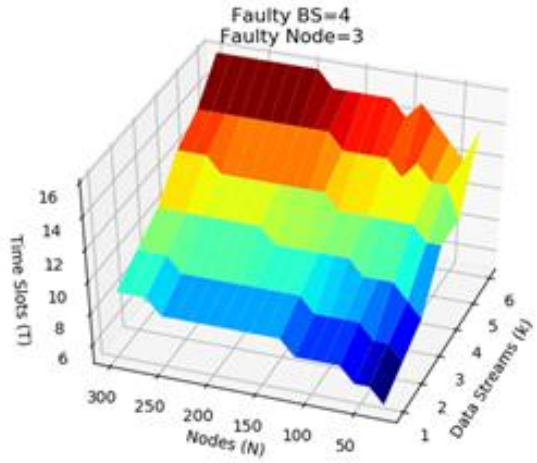


Fig. C(xviii)
Number of nodes (N) = 297
Faulty nodes ($\hat{n}$) = 3
Faulty base station ($\hat{b}$) = 4
Total Data Streams (k) = 6

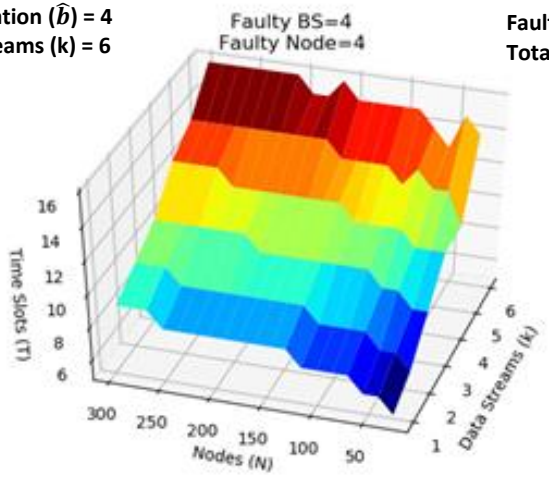


Fig. C(xix)
Number of nodes (N) = 296
Faulty nodes ($\hat{n}$) = 4
Faulty base station ($\hat{b}$) = 4
Total Data Streams (k) = 6

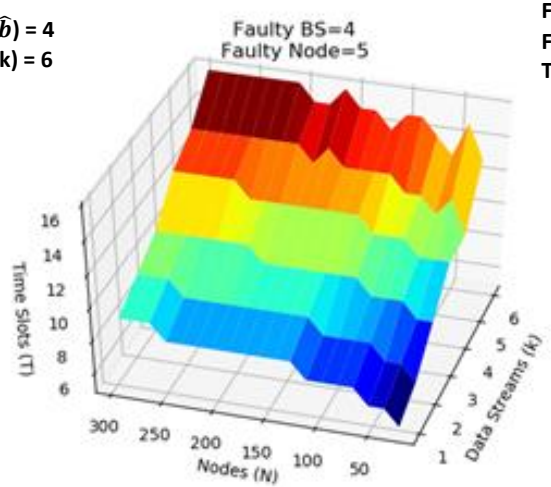


Fig. C(xx)
Number of nodes (N) = 295
Faulty nodes ($\hat{n}$) = 5
Faulty base station ($\hat{b}$) = 4
Total Data Streams (k) = 6

RESULTS— NODES FAULTY & FIVE BS FAULTY

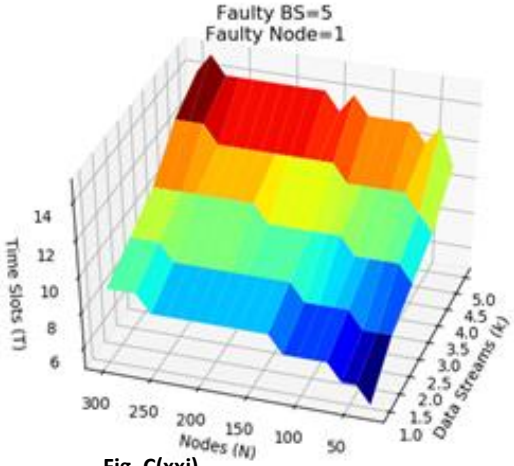


Fig. C(xxi)
Number of nodes (N) = 299
Faulty nodes ($\hat{n}$) = 1
Faulty base station ($\hat{b}$) = 5
Total Data Streams (k) = 5

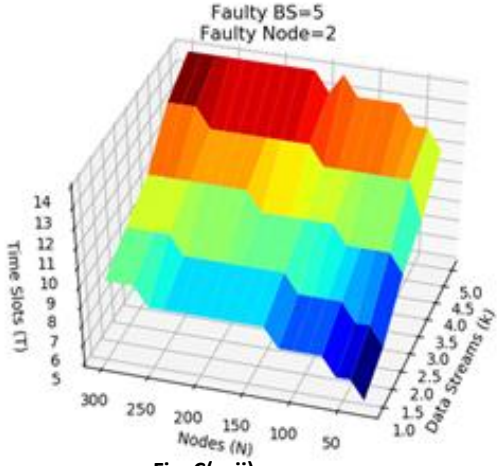


Fig. C(xxii)
Number of nodes (N) = 298
Faulty nodes ($\hat{n}$) = 2
Faulty base station ($\hat{b}$) = 5
Total Data Streams (k) = 5

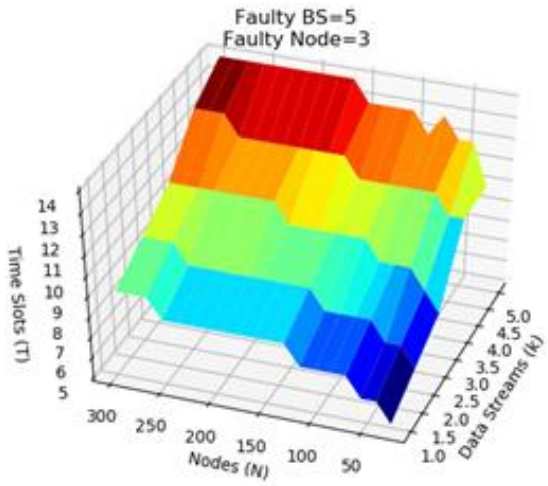


Fig. C(xxiii)
Number of nodes (N) = 297
Faulty nodes ($\hat{n}$) = 3
Faulty base station ($\hat{b}$) = 5
Total Data Streams (k) = 5

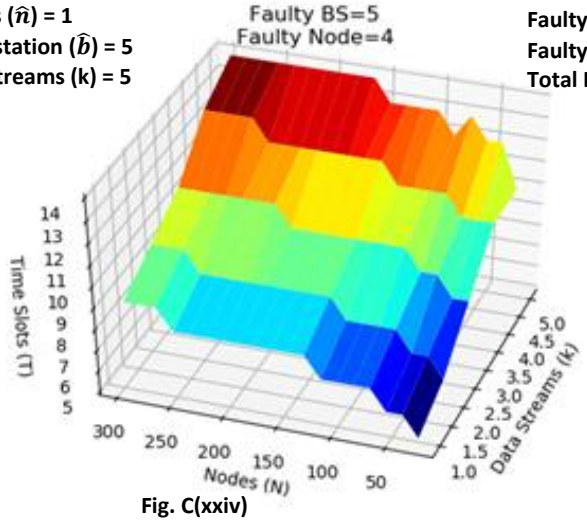


Fig. C(xxiv)
Number of nodes (N) = 296
Faulty nodes ($\hat{n}$) = 4
Faulty base station ($\hat{b}$) = 5
Total Data Streams (k) = 5

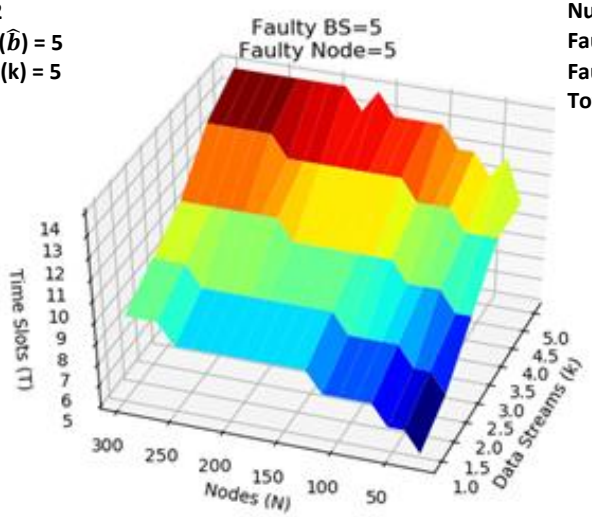


Fig. C(xxv)
Number of nodes (N) = 295
Faulty nodes ($\hat{n}$) = 5
Faulty base station ($\hat{b}$) = 5
Total Data Streams (k) = 5

Chapter V

CONCLUSION

With ever increasing demand of IoT devices it becomes difficult to maintain a network with minimum delay and in real world fault free. Thus in this project we have successfully devised an algorithm for concurrent data collection trees that promise data collection time with minimum delay. This algorithm administers the problem of a faulty node as well as a faulty base station. In the next phase we worked work upon different scenario of fault tolerance in two case i.e alpha ring and beta ring. From these cases we concluded that whether data can be recovered or not and to what extent.

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.

[10] Florens, M. Franceschetti, and R. J. McEliece, "Lower bounds ondata collection time in sensory networks," IEEE J. Sel. Areas Commun.,vol. 22, no. 6, pp. 1110–1120, Aug. 2004.

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

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

[13] Z. Y. Chen and X. F. Wang, "Effects of network structure and routing strategy on network capacity," Phys. Rev. E, vol. 73, no. 3, pp. (036107) 1–5, Mar. 2006.

[14] M. Song and B. He, "Capacity analysis for flat and clustered wireless sensor networks," in Proc. Int. Conf. Wireless Algorithms, Syst. Appl.,(WASA 2007), Chicago, IL, Aug. 2007, pp. 249 -253.

[15] Billionnet A., "Different formulations for solving the heaviest subgraph problem," Inform. Syst. Oper. Res., vol. 43, no. 3, pp. 171–186,Aug. 2005.

[16] Z. Bi, L. D. Xu, and C. Wang, "Internet of things for enterprise systems of modern manufacturing," Industrial Informatics, IEEE Transactions on, vol. 10, no. 2, pp. 1537–1546, 2014.

[17] Y. J. Fan, Y. H. Yin, L. D. Xu, Y. Zeng, and F. Wu, "IoT-based smart rehabilitation system," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1568–1577, May 2014.

[18] L. D. Xu, W. He, and S. Li, "Internet of things in industries: Asurvey," IEEE Transactions on Industrial Informatics, vol. 10,no. 4, pp. 2233–2243, Nov 2014.

[19] S. Fang, L. D. Xu, Y. Zhu, J. Ahati, H. Pei, J. Yan, and Z. Liu,"An integrated system for regional environmental monitoring nd management based on internet of things," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1596–1605,May 2014.

[20] L. Li, S. Li, and S. Zhao, "QoS-aware scheduling of services oriented internet of things," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1497–1505, May 2014.

APPENDIX A

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

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

for $\hat{n}$ nodes

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

II) Case B – When base station becomes faulty

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

Proof:

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

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

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

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

$\hat{n}$: total number of faulty nodes

For $\hat{b}$ base stations

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

III) When $\hat{n}$ nodes and $\hat{b}$ base stations become faulty

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

It can be proved in a similar way as the above two cases.

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

$\hat{n}$: total number of faulty nodes

APPENDIX B

Code for Generating 3D Plots

```
import pandas as pd
importnumpy as np
import math
from mpl_toolkits.mplot3d import axes3d
importmatplotlib.pyplot as plt
frommatplotlib import cm
fromscipy.interpolate import griddata
data=pd.DataFrame()
data1=pd.DataFrame()
n=int(input("Enter number of nodes: "))
k=int(input("Enter value of k: ")) #number of data streams or base
stations
ni=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ii=[]
m=[]
jj=[]
um=[]
alphar=[]
betar=[]
tau1=[]
tau2=[]
T=[]
fori in range(30,n+1,15):
m.append(i)
for j in range(1,k+1):
ii.append(i)
jj.append(j)
umax=math.floor(math.fabs(i)/j)
if(umax==0 or umax==1):
alphar.append('0')
betar.append('0')
tau1.append(1)
tau2.append(0)
T.append(1)
elif(umax%2==0 and umax>=4):
            t1=math.floor((2*(math.fabs(i)-umax)/umax)+1)
            temp2=math.fabs(i)-(t1*(umax/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0
tau1.append(t1)
tau2.append(t2)
```

```
            t=t1+t2
T.append(t)
nalpha=umax/2
alphar.append(nalpha)
betar.append('0')
elif(umax==2):
            t1=math.floor((2*(math.fabs(i)-umax)/umax)+1)
            temp2=math.fabs(i)-(t1*(umax/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0
tau1.append(t1)
tau2.append(t2)
            t=t1+t2
T.append(t)
alphar.append('2')
betar.append('0')
elif(umax==3):
            t1=math.floor((2*(math.fabs(i)-umax)/(umax+1))+1)
            temp2=math.fabs(i)-(t1*((umax+1)/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0
tau1.append(t1)
tau2.append(t2)
            t=t1+t2
T.append(t)
betar.append('1')
alphar.append('0')
elif(umax%2!=0 and umax>=5):
            t1=math.floor((2*(math.fabs(i)-umax)/(umax+1))+1)
            temp2=math.fabs(i)-(t1*((umax+1)/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0
tau1.append(t1)
tau2.append(t2)
            t=t1+t2
T.append(t)
nalpha=(umax-3)/2
alphar.append(nalpha)
betar.append('1')
um.append(umax)
```

```
data['N']=ii
data['k']=jj
data['umax']=um
data['Alpha Rings']=alphar
data['Beta Rings']=betar
data['Tau 1']=tau1
data['Tau 2']=tau2
data['T']=T

data.to_csv('col1.csv', index=False)
fig = plt.figure()
ax = fig.gca(projection = '3d')
X=np.array(ni)
Y=np.array(T)
Z=np.array(m)
Y=np.reshape(Y, (19,10))
x, z= np.meshgrid(X, Z)

surf=ax.plot_surface(x, z, Y, color=None,cmap=cm.jet)
ax.set_xlabel('Data Streams (k)')
ax.set_ylabel('Nodes (N)')
ax.set_zlabel('Time Slots (T)')
plt.show()
```

Code for Calculating Time Slots and Data Streams

Alpha Ring

```
import math
hd=[]
c1=[]
c2=[]
n=int(input("Enter number of nodes: "))
k=int(input("Enter value of k: ")) #number of data streams or base
stations
i=n
j=k
umax=math.floor(math.fabs(i)/j)

if(umax%2==0 and umax>=4):
            t1=math.floor((2*(math.fabs(i)-umax)/umax)+1)
            temp2=math.fabs(i)-(t1*(umax/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
```

```
                t2=0

            t=t1+t2

nalpha=umax/2

elif(umax==2):
            t1=math.floor((2*(math.fabs(i)-umax)/umax)+1)
            temp2=math.fabs(i)-(t1*(umax/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0

            t=t1+t2

elif(umax==3):
            t1=math.floor((2*(math.fabs(i)-umax)/(umax+1))+1)
            temp2=math.fabs(i)-(t1*((umax+1)/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0

            t=t1+t2
elif(umax%2!=0 and umax>=5):
            t1=math.floor((2*(math.fabs(i)-umax)/(umax+1))+1)
            temp2=math.fabs(i)-(t1*((umax+1)/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0

            t=t1+t2

nalpha=(umax-3)/2



l=64
fori in range(1,k+1):
    l=l+1
for j in range(1,t1+1):
hd.append(chr(l)+str(j))
cone=int(math.fmod((2*(i-1)+(j-1)),n)+1)
ctwo=int(math.fmod((2*(i-1)+(j)),n)+1)
c1.append(cone)
```

```
c2.append(ctwo)
    #hd.append('|')
c1.append('|')
c2.append('|')

print(hd)
print(c1)
print(c2)
```

Beta Ring

```
import math
c3=[]
c4=[]
c5=[]
hd=[]
n=int(input("Enter number of nodes: "))
k=int(input("Enter value of k: ")) #number of data streams or base
stations
i=n
j=k
umax=math.floor(math.fabs(i)/j)

if(umax%2==0 and umax>=4):
            t1=math.floor((2*(math.fabs(i)-umax)/umax)+1)
            temp2=math.fabs(i)-(t1*(umax/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0

            t=t1+t2

nalpha=umax/2

elif(umax==2):
            t1=math.floor((2*(math.fabs(i)-umax)/umax)+1)
            temp2=math.fabs(i)-(t1*(umax/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0

            t=t1+t2

elif(umax==3):
```

```
            t1=math.floor((2*(math.fabs(i)-umax)/(umax+1))+1)
            temp2=math.fabs(i)-(t1*((umax+1)/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0

            t=t1+t2
elif(umax%2!=0 and umax>=5):
            t1=math.floor((2*(math.fabs(i)-umax)/(umax+1))+1)
            temp2=math.fabs(i)-(t1*((umax+1)/2))
if(temp2>0):
                t2=math.floor(math.log2(temp2))+1
else:
                t2=0

            t=t1+t2

nalpha=(umax-3)/2



l=64
fori in range(1,k+1):
    l=l+1
for j in range(1,t1+1):
hd.append(chr(l)+str(j))
cthree=int(math.fmod((3*(i-1)+2*(j-1)),n)+1)
cfour=int(math.fmod((3*(i-1)+2*(j-1)+1),n)+1)
cfive=int(math.fmod((3*(i-1)+2*(j-1)+2),n)+1)
c3.append(cthree)
c4.append(cfour)
c5.append(cfive)
c3.append('|')
c4.append('|')
c5.append('|')

print(hd)
print(c3)
print(c4)
print(c5)
```