DATA COLLECTION NETWORK STRUCTURE TREE FOR IoT

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

in

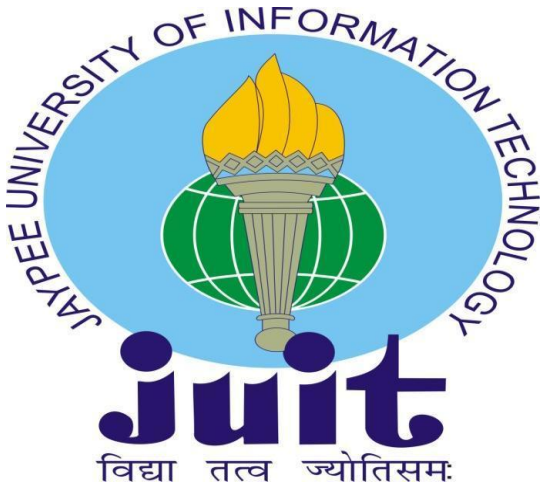
Computer Science and Engineering/Information Technology

By

Kaushik Deka(191378) & Adarsh Kumar (191404)

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

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Data collection network structure tree for IoT"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2022 to December 2022 under the supervision of **Mr. Arvind Kumar, Assistant Professor, CSE/IT**.

I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Cloud Computing.**

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

Kaushik Deka(191378)

Adarsh Kumar (191404)

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

Computer Science & Engineering and Information Technology

Dated: 28. 11. 22

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

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

I would like to express my heartiest gratitude to **Mr. Arvind Kumar,** Department of CSE & IT, for her kind help to finish my project.

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

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

Student's Name : Kaushik Deka
Roll No. 191378

Student's Name : Adarsh Kumar
Roll No. 191404

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviations	**Meaning**
IoT	Internet Of Things
TDMA	Time Division Multiple Access
WSN	Wireless Sensor Network
CDCT	Concurrent Data Collection Trees
MAC	Media Access Control
MC2H	Multiple Cluster 2 Hop
BS	Base Station
N2N	Node To Node Transmission
N2BS	Node To Base Station Transmission
TOCDCT	Time Optimal Concurrent Data Collection Tree
DADCNS	Delay Aware Data Collection Network Structure

LIST OF FIGURES

LIST OF GRAPHS

LIST OF TABLES

ABSTRACT

Typically, an Internet of things (IoT) application consists of a number of smart devices that produce and exchange enormous volumes of data. Several applications can work together and share the same hardware infrastructure to satisfy their individual sensing requirements. Another advantage of such a shared network setup is that it benefits several subscribers to the same data. To boost productivity, the data that these devices produce is examined. Additionally, it is employed to enhance security and safety. A typical Internet of Things (IoT) network consists of several hundred interconnected devices, and numerous application processes rely on the data these devices produce. These applications collaborate and utilize the same device infrastructure to fulfill their individual sensing requirements in order to avoid over provisioning. However, this creates the issue of concurrent data collection. Multiple parallel data streams can be employed in concurrent data collecting methods to gather data effectively at numerous base stations. For IoT applications, the concurrent data gathering trees' present designs pose a number of additional difficulties. The delay optimization of the concurrent data collection procedures is one such difficulty. A Fast concurrent data collecting tree is suggested in this research. We demonstrate through simulations that the proposed design outperforms the current approach in terms of data collecting speed.

1. INTRODUCTION

1.1 Introduction

70% of the world's population is projected to reside in cities by 2050. Cities must use contemporary information and communications technologies to quickly offer current information to their citizens in order to enable such rapid growth Internet of Things (IoT) is one of the technologies that is widely acknowledged as a viable solution. The majority of smart cities now in existence have non-interoperable, standalone IoT infrastructures. Instead of building numerous distinct closed form systems, IoT devices put on various assets should be interconnected in order to maximize efficiency and unlock the full potential of smart cities.

Additionally, IoT infrastructures in the public and private sectors should be shared to prevent overprovision and pointless redundancy. A smart city, for instance, often consists of numerous different applications, Hence different applications can be interested in the data produced by the same group of devices. Additionally, it's possible that different programmes may use the same data in different ways. Applications in such a situation stand to gain greatly from the shared infrastructure's reduced deployment costs, time, and maintenance requirements. Furthermore, using the infrastructure-as-a-service business model, similar infrastructure can also be made available to apps for a fee. Data collecting, privacy and data security, resource distribution, energy use, and a few other issues need to be addressed. Since numerous applications may query data sources at once and since the freshness of the data must be maintained, the process of data collection presents a significant difficulty.

1.2 Problem Statement

The concurrent data collection trees are designed to collect the data parallelly from the same set of nodes to various base stations by minimizing the delay. The network structure proposed in

Delay Aware Data collection Network Structure Tree (DADCNS) [1] is shown to be sub-optimal in terms of the number of required time-slots as well as the limitations of the network to only number of nodes in the value N = 2^p where p = 1, 2, 3,..... With a higher number of nodes that can be utilized in a time-slot, fewer time-slots would be required for data collection. In this Project, a delay aware time optimal concurrent data collection trees are proposed to keep the overall data collection duration short. Simulation findings demonstrate that the suggested concept can significantly cut down on delays in concurrent data collection processes..

1.3 Objectives

The main objective here is to optimize the time available for data collection. The more fresh the data is if it is supplied as quickly as it is collected. As a result, data must be gathered simultaneously at several stages of data extraction. The issue of concurrent data gathering in the IoT has only been addressed in a small number of recent research in this area.

Here, maximizing the amount of time spent collecting data is the main driving force. The more fresh the data is if it is delivered as quickly as it is collected. As a result, data must be gathered simultaneously at various points of data extraction. The issue of concurrent data collection in the IoT has only been addressed in a small number of recent works in this area.Two such algorithms that use various topological structures to deliver data concurrently are presented in [2] and [4] specifically.

In order to speed up the concurrent data collection processes, we further optimize the network topology in this paper. According to the results of the simulation, the proposed structure for concurrent data collection can reduce the delay.One such approach that leverages several topological structures to transmit data concurrently is presented in detail. In order to speed up

the concurrent data gathering procedures, we further optimize the network design in this project. According to the results of the simulation, the proposed structure for concurrent data collecting can reduce the delay.

1.4 Methodology

The concurrent data collection trees are intended to minimize the delay when collecting data concurrently from the same set of nodes to various base stations. The proposed network structure is a tree structure. We can have any number of Nodes and Base stations and our model should provide a definite structure to ensure fast concurrent data collection from all the nodes. Nodes are sensors that collect information and data from the environments and have the computational power to store and forward that data to other nodes as well as base stations. Base stations are fixed transceivers that are the main communication point for one or more IoT devices. The process of data collection is an important challenge as multiple applications and processes may simultaneously absorb data sources, and as freshness of data has to be maintained, there is a need for concurrent data collection techniques.

Our proposed structure is a tree structure of a number of clusters. These clusters are of the size 2^p where p=1,2,3…. For any number of inputs lets say N we divide the N nodes into different clusters of size 2^p. The clusters internally arrange themselves using the DADCNS algorithm. The model will form a hierarchical structure where the cluster members will send data to their respective parent nodes. Similarly, the parent nodes will send the aggregated data from its child nodes to the cluster heads. The cluster heads will then aggregate the collected data and send it to the base stations.

1.5 Organization

The First chapter provides a brief introduction to the proposed problem and how we are planning to solve the problem. It also provides a brief overview of the current IoT domain and how fast data collection from sensors is extremely important in achieving zero delay real time information to various applications used in cloud or otherwise. The problem statement and the proposed methodology is briefly discussed in the chapter.

The Second chapter provides an overview of the various works on similar domain that was carried out previously by various authors. It was seen that various types of modes were proposed using different algorithms and solutions to reduce the data collection speed of IoT devices as well as WSNs. The issue of concurrent data collection in the IoT has only been addressed in a small number of recent works in this area. One such algorithm that uses various topological structures to deliver data concurrently is presented in [2] specifically. In order to speed up the concurrent data collection processes, we further optimize the network topology in this paper. According to the results of the simulation, the proposed structure for concurrent data collection can reduce the delay.

The third chapter gives an overview of the proposed model. How the model is specifically and mathematically designed. The methodology of the proposed is provided in this section. The sensor devices in WSNs are often deployed to address a particular demand or purpose. In IoT networks, on the other hand, the devices can be shared between many parties or applications, which may cause data collecting to occur simultaneously on the same set of devices. As a result, thinking about concurrent data collection at various base stations becomes necessary. Results from [2] demonstrated that for a single data collection process, concurrent data collection trees (CDCT) have a shorter data collection duration than the current data collection network structure.

The fourth chapter provides the analysis of the algorithms of the proposed model using examples. Various simulations were performed on different formats of inputs. Firstly the

number of Nodes we varied and the number of base stations is kept constant. In the second case, the opposite is done. In both the scenarios the resultant timeslots are compared to CDCT and Time Optimal CDCT.

The Fifth and final chapter provides a summary of the entire project and what we can conclude from the simulations and observation. Fast data collection from IoT sensors can help receive data in lesser delay form ever before. which can eventually help benefit globally since IoT has become a day to day application in our lives. This project could be extended to various domains. For this particular research, we have only looked at the structure of the network between the nodes and the base stations. In future we can also look into the frequency of each device and the energy consumption of these data collection processes and try to minimize it. This could in turn help us to reduce the use of energy to operate an IoT network. The future scope of the proposed project and its various applications are also provided in this chapter.

Chapter-2 Literature Survey

We have explored fast data collecting with the aim of reducing the aggregated data collection schedule length in [3, 9], and others have studied it in [11, 14,16]. While the theoretical parts were covered in [3], where we suggested constant factor and logarithmic approximation methods on geometric networks, we empirically examined the effects of transmission power regulation and various frequency channels on the schedule length in [3]. (disk graphs). A distributed time slot assignment approach is suggested by Gandham et al. [15] to reduce the TDMA schedule length for a single channel. Raw-data convergecast has been examined in [15]. Moscibroda [16] investigates the topic of joint scheduling and transmission power regulation for constant and uniform traffic needs. In contrast to the work mentioned above, ours evaluates transmission power regulation in real-world scenarios and computes lower constraints on the schedule length for tree networks using algorithms to reach these bounds. We also compare the effectiveness of various interference models and channel assignment techniques, and we suggest strategies for building particular routing tree topologies that increase the data collecting rate for both aggregated and raw-data convergecast. By allocating time slots to nodes from the bottom of the tree to the top, orthogonal codes have been examined as a means of reducing interference. This ensures that a parent node does not transmit before it has received all of the packets from its offspring. This issue as well as the one that Chen et al. [17] refers to converge casts of one-shot raw data. Because the number of hops in a tree increases as its degree decreases, the degree-constrained routing topologies we create in this study may not always result in schedules with minimal latency. Therefore, additional optimization, such as building bounded-degree, bounded-diameter trees, is required if decreasing latency is also a necessity. Pan and Tseng [17] provide a study along these lines with the goal of minimizing the maximum latency in which they allocate each node in a Zigbee network a beacon period during which it can receive data from all of its offspring. Song et al. [13] introduced a time- and energy-efficient packet scheduling technique with periodic scheduling for raw-data convergecast. However, they briefly discuss a 3-coloring channel assignment mechanism, and it is unclear whether the channels are frequencies, codes, or some other method to eliminate interference. Once interference is eliminated, their algorithm achieves the bound that we

describe here. Additionally, they make the assumption that there will be no cumulative interference from contemporaneous multiple senders and that each node will have a circular transmission range. In contrast to their work, we take into account multiple frequencies,assess the performance of three different channel assignment techniques, and assess the effects of transmission power control using realistic interference and channel models, such as a physical interference model and overlapping channels, while also taking routing topologies into consideration. Song et al. [13] expanded on their work and suggested a MAC protocol based on TDMA. In order to reduce congestion, TreeMAC takes into account the variations in load at different levels of a routing tree and assigns time slots based on the depth, or the number of hops, of the nodes on the routing tree. This means that nodes closer to the sink are given more slots than their children. However, because the sink can receive every three time slots, TreeMAC operates on a single channel and achieves 1/3 of the maximal throughput, similar to the constraints stated by Gandham et al. [15]. Choi et al. [17] demonstrate that the challenge of minimizing the schedule length for raw-data convergecast on a single channel is NP-complete on general graphs. Lai's research focuses on increasing convergecast throughput by locating the shortest-length, conflict-free schedule. where the senders are given time slots by a greedy graph coloring technique to avoid interference. They also talked about how routing trees affect schedule length and suggested a routing technique called disjoint strips to send data along many shortest paths. Sending data across other pathways does not shorten the schedule, however, because the sink continues to be the bottleneck. As we shall demonstrate in this work, utilizing capacitated minimum spanning trees for raw-data convergecast—where the number of nodes in a subtree is no more than half the total number of nodes in the remaining subtrees—provides the routing structure with an advantage. Both cellular and ad hoc networks have undergone substantial research into the utilization of various frequencies, However, there are a few studies that make use of several channels in the field of WSN. In order to achieve this, we assess the effectiveness of three distinct methods that handle the channel assignment at various levels.

Energy conservation becomes one of the main problems in sensor networks as a result of the energy limitations of individual sensor nodes. In wireless sensor networks, wireless communications account for a sizable amount of a node's energy consumption. A transmission's energy consumption is inversely correlated with the communication distance. As a result, base

station and node long-distance communication is typically discouraged. Adopting a clustering method is one way to reduce energy usage in sensor networks [5]. A method called clustering aims to group sensor nodes into clusters. One node is chosen to serve as the cluster head inside each cluster. The cluster leader is in charge of gathering data from the cluster's members and fusing it using data/decision fusion techniques. and 3) reporting the fused data to a remote base station The sole node in each cluster that engages in long-distance communication is the cluster head. As a result, the network as a whole uses less energy. There has been a lot of research done on how to construct clusters with the right network architecture to save energy [6, [7, [8,] [9]. LEACH is a clustering algorithm that Heinzelman et al. devised [6]. In LEACH networks, sensor nodes are arranged in multiple-cluster 2-hop (MC2H) networks (basis station cluster head cluster members). The number of long-distance transmissions can be significantly decreased by using the clustering concept. Another clustering algorithm by the name of PEGASIS [7], put forth by Lindsey and Raghavendra, takes an entirely different approach by grouping sensor nodes into a single-chain (SC) network. In these networks, only one chain node is chosen as cluster head. The amount of energy used in long-distance transmission is further reduced by reducing the number of cluster heads. PEDAP [8], created by Tan and K. Orpeoglu, is based on the concept of a minimal spanning tree (MST). The communication distances between sensor nodes are reduced, in addition to the amount of long-distance transmission. The collection tree protocol (CTP) was proposed by Fonseca et al. [9]. Expected transmissions (ETX) serve as the routing gradient in the CTP, a type of gradient-based routing system. The anticipated transmissions (ETX) of a packet are the minimum required for error-free reception [6]. High throughput is anticipated for paths with low ETX. In a CTP network, nodes always choose the route with the lowest ETX. The ETX of a path is typically inversely correlated with the associated path length [18]. As a result, CTP can significantly shorten the communication gaps between sensor nodes. All of these algorithms have potential energy-saving benefits. A network created by an energy-efficient clustering technique, however, would not always be preferred for data collecting. In Section V, a performance analysis of these network architectures' data gathering effectiveness will be presented. The goal of this research is to examine the effectiveness of data collecting in networks created by various clustering methods. Event triggering algorithms like TEEN and APTEEN won't be taken into account in this research. The related study on effective data gathering was completed by Florens et al. [10].

Lower constraints on data collecting time for different network structures are derived in their work. The impact of data fusion, which is thought to be one of the key characteristics of sensor networks, was not taken into account. For wireless sensor networks, Wang et al[11] .'s suggested link scheduling algorithms that can significantly increase network throughput. However, it is assumed in their work that data linkages between wireless sensor nodes are predefined. In contrast, the goal of this paper is to create data linkages between wireless sensor nodes in order to reduce the time used for data gathering. Solis and Obraczka completed another comparable piece of work in which they investigated the effects of timing on data aggregation for sensor networks. Chen et al. [17] looked into the effects of network capacity under different network capacity under different network structure and routing algorithms. According to their research, capacity refers to the amount of end-to-end traffic that a network can support. Their main worry is not a delay in the data collection procedure.

The early work of Cheng et al. [17] examined the issue of data collecting in large-scale sensory systems. In their research, they took into account gathering data from a sizable number of people to a single data extraction point. The authors of [17] have offered a fresh perspective on the well-studied routing problem in computer networks based on the observation that all the nodes in a sensory system are owned by the same party. That is, one should optimize the use of his network resources by establishing a coordinated transmission schedule rather than avoiding crowded channels. A framework for assessing the time performance of data collection and data distribution tasks in sensory systems was offered by Florens et al. in [10]. When they work, For networks with different topologies, they deduced low bounds and provided the matching optimal transmission schedule. The first researchers to focus on the issue of continuous data collection in sensory systems are Ji et al. They developed the lower bounds for single-snapshot data collections and continuous data collections in their work. They also demonstrated how the use of devices with numerous transceivers may considerably speed up a data collection operation. The aforementioned efforts served as the basis for [10], [11]'s creation of delay-aware data gathering network topologies. A delay-minimized network topology for fusible data and its accompanying construction methods for centralized and distributed systems were introduced by Cheng et al. in [17]. Another network topology was introduced by Cheng et

al. in [17] to enable opportunistic in-network data fusion. which a star network's top bound can never surpass. A delay-aware network topology and its construction algorithm have been presented in to overcome conflicts between transmission schedules for sensory systems that demand sequential data gathering operations. Chen et al analysis of the data gathering issue in [17] took channel models into account. They established upper and lower limitations for data collection techniques in networks where data fusion is not appropriate in their work. A quick data aggregation tree for single-snapshot data collecting in wireless sensor networks was put forth by Durmaz Incel et al. in [3]. Interferences between sensor nodes are considered throughout the tree construction process. Wang et alstrategy .'is involved getting an By choosing only some of the nodes for sampling, approximately obtaining data. For instances where data demonstrate a high degree of spatial correlation, their suggested solution is quite effective and dependable. Optimizing transmission schedules in sensory systems with dynamic traffic patterns has recently been taken into consideration by works in [19]. In [20] and [21], probabilistic network models for sensory systems were studied. [18] has researched the data collection procedures in sensory systems with mobility. However, the job scheduling problem in wireless sensor networks (WSNs) was examined in the works by Kapoor et al. which may have the most similarities to the issue under consideration in this work. It is emphasized that the main enabling technology that distinguishes the Internet of Things (IoT) from traditional sensory systems is machine-to-machine (M2M) communication. Sensor nodes in typical WSNs are typically owned and operated by a single entity. However, in IoT applications, IoT devices might be shared by a number of users or programmes, who can then start numerous concurrent data aggregations on the same set of nodes. Unfortunately, none of the works mentioned above take into account numerous data extraction sites and concurrent data collection operations. Recent years have seen a lot of interest in data collection in real-world IoT systems that a global-scaled IoT federation may be achieved by using satellite data lines to link apart disparate IoT fragments. Although an average packet loss rate of about 25% is anticipated in IoT systems, delays brought on by retransmissions can be minimized by compressing data and eliminating packet fragmentations.Wu et al. claimed that when devices are sharing a single channel, data gathering systems based on the conventional IEEE 802.11 standard may experience performance degradations. For effective data gathering in massive IoT systems, they presented an adaptive channel allocation method and an energy-aware access control protocol.

By fusing mobile ad hoc networks (MANETs) with WSNs, Bellavista et al. are the first in the field to introduce mobility into IoT [22]. The primary goal of the data collection algorithms created for wireless sensor networks (WSNs) was to extend the network lifetime by sacrificing data collection efficiency.In some ways, a routing protocol is akin to a data-collection procedure because it seeks to create a route that nodes can follow to convey data to an extraction point.In [5], a list of these routing algorithms is shown. As an alternative,[6]–[7] studies concentrated on reducing energy use by establishing clusters. For data collecting, various tree topologies including chain, minimum spanning, and cluster trees were taken into consideration. On various topologies and network structures, Florens et al. [10] estimated a lower bound for the data collecting time after analyzing the effectiveness of data gathering. The effect of time on data aggregation for sensor networks is discussed in [9] and [10]. By reducing communication costs, Yin et al. introduced a data aggregation tree for sophisticated searches.

To maximize the aggregate gain, they took into account both the data pruning power and the aggregation cost in their work. A data gathering architecture in the Internet of Mobile Things as a service platform was addressed by Maiti et al. in [23]. They discussed how data gathering relates to the quantity of sensors, energy management, data privacy, and data quality in this work. For multi-hop, device-to-device (D2D) communication with decode and forward relaying, Chen et al. [17] established an optimal routing based on trustworthy connectivity probability by taking fixed and random locations of base stations into consideration. Using a coalition formation game, Xu et al. created a multi-hop routing strategy to minimise network delay. It is demonstrated that the game offers a Nash-stable equilibrium. In [17], Chen et al. used a Poisson distribution to investigate the performance of multi-hop routing. At a single base-station, Kolcun et al. presented a distributed data gathering mechanism in [24]. In order to provide speedier data collecting by selecting alternate channels for data transfer in the event of network congestion, Cheng et al. [17] introduced a coordinated data collection system. Ji et al[20] .'s continuous data collection method relied on making use of all available network resources. A network layout that divides the nodes into clusters of varying sizes and allows the nodes in each cluster to alternately transmit to the base station is proposed by Cheng et al. in [17]. To speed up the data collection process, they have further optimized the inter-cluster communication distances.

CONCURRENT DATA COLLECTION TREES

Consider an Internet of Things (IoT) network N = n1, n2, , n|N| and a collection of base stations S = s1, s2, , s|S|. It is expected that each of these |N| IoT nodes can connect to the base stations and communicate with one another. Since numerous incoming data packets can be fused into one before being forwarded to one's parent node, data collected from various IoT devices is believed to be perfectly fusible [6]. A single unit of data will be transmitted throughout one time slot, and it is believed that the time it takes to combine the data will be minimal. The total number of concurrent data streams is k, and each concurrent data aggregation process will access the IoT network via a distinct base station (BS).

The issue of concurrent data collecting using several data streams at multiple base stations is first addressed by the authors in Concurrent Data Collection Trees. The aforementioned method is based on the design of CDCT, which is depicted as rings, also known as α-rings and β-rings. The data collecting time from the same set of nodes to many base stations is shortened as a result of this network structure. The network structure is predicated on the idea that nodes can combine data from many IoT nodes into a single packet before sending it on. It is anticipated that a single piece of data will be conveyed once. Such data aggregation operations can all operate simultaneously to a different IoT base station. The total amount of base stations is equivalent to the total amount of active data gathering operations, which is indicated by the letter k. Additionally, it is believed that the network's transmissions are synced. In other words, numerous transmissions can take place simultaneously between non-overlapping sets of nodes. The proposed network configuration makes use of the most nodes possible during each time slot, depending on the quantity of nodes and data streams. Additionally, each data stream inside a time slot must use a unique combination of nodes for transmission. Umax specifies the maximum number of nodes that can be used by a data stream in its first time slot.

All concurrent data streams should start and stop during the same time period in order to guarantee fairness among these users. However, each time slot in parallel data streams should use the same number of nodes. Each data stream should make use of the greatest number of nodes at each time-slot in order to accelerate the overall data collection process.

TIME OPTIMAL CONCURRENT DATA COLLECTION TREES

In contrast to Wireless Sensor Networks (WSNs), the Internet of Things (IoT) allows many applications to share the same device infrastructure. The devices can be queried by multiple such apps at once, which might necessitate starting concurrent data streams on the devices. The authors have noted this problem in [2]. In order to overcome this, they presented a concurrent data gathering tree structure known as α-rings and β-rings that is represented as rings. These rings are used to create the network architecture, and data is collected simultaneously at several base stations (BSs) using the same set of nodes. Therefore, it's crucial to make the most of the nodes throughout a particular time period. To do this, [4] concentrated on the β-rings rather than the α-rings to maximize node utilization.

The Time Optimal CDCT network structure that minimizes the quantity of time slots needed for concurrent data collecting is defined here. The network configuration consists of a number of devices or nodes, denoted by N = {n1, n2, . . . , n|N|} . . base-stations (BS) S = {s1, s2, . . . , s|S|} are used to symbolize, n|N| that are shared by many applications. s1,s2,s3 . . , s|S|. We take into account a single-hop network architecture in which devices can connect directly to the BS. A device can also talk to any other device in its vicinity. Assuming that the data produced by these devices is related, devices can combine and communicate data. Concurrent data collection is necessary because multiple applications may need data at once.. All devices in N are assumed to have some data to send, and are involved in data transmission. It is assumed that every device in the network N is engaged in data transmission and has some data to send. A certain number of concurrent datastreams k are started in the network depending on how many of these parallel applications are requesting data. Each time-slot is typically thought of as transmitting one unit of data. The suggested network topology makes use of the most devices in the first τ1 time-slots, depending on the quantity of devices and data streams. The devices used for data transmission of various data streams vary depending on the time slot. Umax specifies the most devices that can be used by a data stream in the first time slot

Chapter-3 SYSTEM DEVELOPMENT

The proposed network structure is a tree structure. We can have any number of Nodes and Base stations and our model should provide a definite structure to ensure fast concurrent data collection from all the nodes. Nodes are sensors that collect information and data from the environments and have the computational power to store and forward that data to other nodes as well as base stations. Base stations are fixed transceivers that are the main communication point for one or more IoT devices. The process of data collection is an important challenge as multiple applications and processes may simultaneously absorb data sources, and as freshness of data has to be maintained, there is a need for concurrent data collection techniques.

3.1 INTRODUCTION

We define the network design that optimizes the number of time-slots required for concurrent data collection. The network architecture consists of a set of devices/nodes, represented by N = {n1, n2, . . . , n|N|} that are shared among multiple applications, represented by set of base-stations (BS) K = {k1, k2, . . . , k|K|}. We consider a multihop network structure, where devices can reach the BS through their cluster head(CH) only. Each device can only communicate with a parent device only. Devices can aggregate and transmit data because it is assumed that the data these devices produce is correlated.

Concurrent data collection is necessary because multiple applications may need data at once. It is assumed that every device in the network N is engaged in data transmission and has some data to send. Initiating such a number of concurrent data streams in the network depends on how many of these parallel applications are requesting data. Each time-slot is typically thought of as transmitting one unit of data. The suggested network topology makes use of the most devices in T time-slots, depending on the quantity of devices and data streams. The devices used for data transmission of various data streams vary depending on the time slot.

Since multiple received data packets can be fused into one before being forwarded to one's parent node, data collected from various IoT devices is assumed to be perfectly fusible . A single unit of data will be transmitted over one time slot, and it is assumed that the time it takes to combine the data will be minimal. All concurrent data streams should start and end during the same time period in order to maintain fairness among these users. However, each time slot in parallel data streams should use the same number of nodes. Each data stream should make use of the maximum possible number of nodes at each time-slot in order for the optimal utilization of the overall data collection process.

3.2 DELAY AWARE DATA COLLECTION NETWORK STRUCTURE TREE(DADCNS)

Before going into our model development, we need to understand DADCNS. This is because we used DADCNS to form the various clusters in our model. We have overcome a specific weakness of this model. This will be later discussed.

DADCNS is a multi hop tree structure. The number of nodes N in the network structure must be limited to N = 2^p, where p = 1, 2, ….. in order to deliver the highest data collection efficiency. A rank, which is an integer between 1 and p, will be assigned to each member of the cluster. A node of rank k will create k-1 data links with other nodes of rank k-1, where k-1 nodes range in rank from 1, 2,..., k-1. The node with rank k will have all of these k- 1 nodes as children. A data link will be created between a node with rank k and a node with a higher rank. The node with rank k will have this higher rank node as its parent node. The cluster head will be treated as an exception. The network node with the highest rank is the cluster head. The cluster head will create a data link with the base station rather than a node of higher rank. By applying this reasoning, Table 3.1 illustrates how the rank distribution will follow an inverse exponential base-2 function.

TABLE 3.1

CLUSTER MEMBERS' RANK DISTRIBUTION IN DADCNS STRUCTURE WITH NETWORK SIZE N = 2^p, WHERE p = 1, 2, · · · .

Rank	1	2	· · ·	$\log_2 N - 1$	$\log_2 N$
No. of nodes	$\frac{N}{2^1}$	$\frac{N}{2^2}$	· · ·	$\frac{N}{2^{(\log_2 N - 1)}}$	$\frac{N}{2^{(\log_2 N)}}$

.

Fig 3.1 displays an example of the DADCNS network with a size of 16. In this illustration, the base station needs 5 Time Slots to gather all of the data from the 16 nodes.

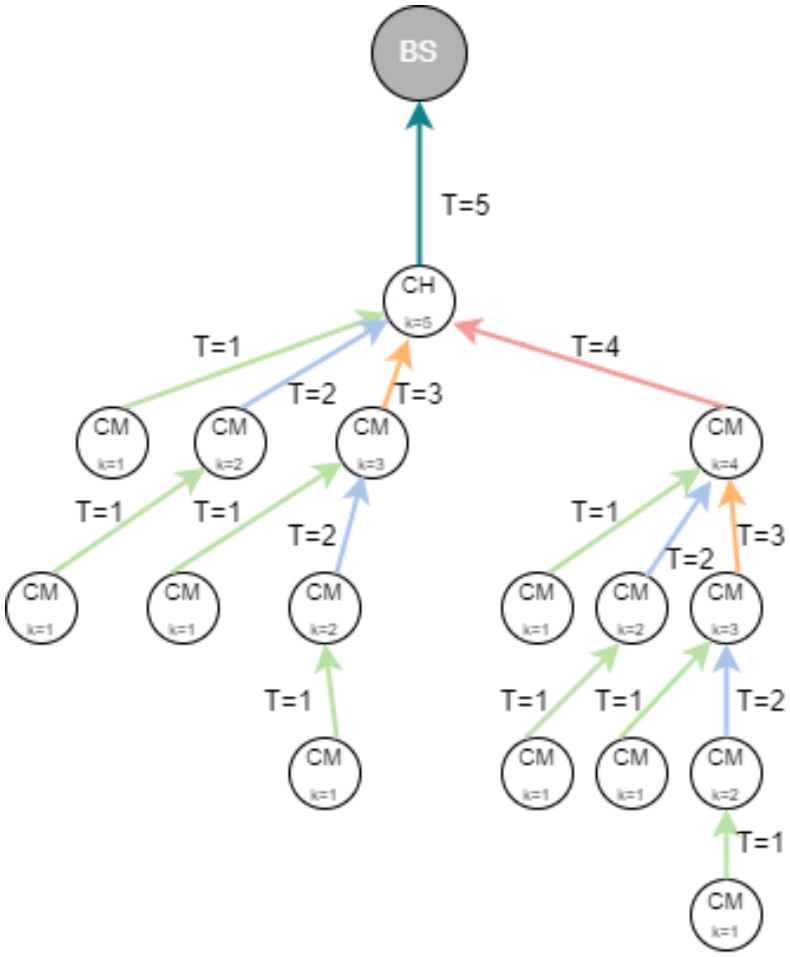


Fig. 3.1. This represents the structure inside one cluster of network size N = 16.The nodes of the cluster are arranged using DADCNS. The cluster's members are shown as circles with CM. The cluster head is represented by a circle with CH. The base station is represented by a filled circle with BS. The variable k

represents the rank of each node. The existence of a data link is represented by a dashed arrow, and the direction of the arrow indicates the direction of data flow.

By dividing the time quantum into time slots of durations T, the above process will take 5 time slots.

3.3 LIMITATIONS OF DADCNS

The main limitation of DADCNS is that this network architecture can work only against a number of nodes in the form of N= 2^p where p =1,2,3…. This makes the mentioned algorithm very restrictive. In the real world the number of nodes will most likely be not in the form of 2^p. So the proposed algorithm in the project is made to remove those limitations as well as perform efficiently in gathering data from any number of nodes and forward it to any number of base stations. The proposed model is explained below.

3.4 PROPOSED DATA COLLECTION TREE STRUCTURE

Our proposed structure is a tree structure of a number of clusters. These clusters are of the size 2^p where p=1,2,3…. For any number of inputs lets say N we divide the N nodes into different clusters of size 2^p. The clusters internally arrange themselves using the DADCNS algorithm. The model will form a hierarchical structure where the cluster members will send data to their respective parent nodes. Similarly, the parent nodes will send the aggregated data from its child nodes to the cluster heads. The cluster heads will then aggregate the collected data and send it to the base stations.

The base stations are numbered serially based on a number of factors. The cluster heads will access the base station serially.

Take a look at a cluster where N = 2^p, where p = 1, 2,...,... The number of time slots needed for a node of rank k = 2 to gather data from all of its child nodes is equal to the number of child nodes it has, which is 1. The case for k = 2 is thus supported. Let's now assume that each node in a connection with k = n needs n- 1 time slots to gather all the data from its child nodes. Node I has n directly connected child nodes, and its rank is k = n + 1. The ranks of each of these directly connected child nodes range from 1 to n.

Since data packets produced by sensor nodes are considered to be highly correlated, a node is always capable of combining all packets it receives using data/decision fusion techniques into a single packet.

Using DADCNS, for each cluster with G nodes with G= 2^p where p=1,2,3…., the cluster head is the only node with the highest ranking which is

$$kmax = \log2(G) + 1 \qquad (1)$$

Similarly, the number of time slots t(G) required for a cluster head, with rank kmax, to collect data from all its child nodes is

$$t(G) = kmax - 1 = \log2(G) \qquad (2)$$

Therefore, the total number of time slots required by one cluster head to send data to the 1st base station is,

$$t(G) = \log2(G) + 1 \qquad (3)$$

By adopting the proposed network structure, the number of time slots T(N) required for K number of base stations to collect data from the whole network with N nodes is given by

$$T(N) = floor(\log2(N)) + K \qquad (4)$$

Or

$$T(N)=M+K \qquad (5)$$

Where M is the number of clusters.

Figure 3.2 shows a network with N devices which has been divided into clusters of 16 nodes, 8 nodes, 4 nodes and 1 node respectively. All these 4 clusters follow DADCNS algorithm internally with ranking each node and its subsequent child nodes.

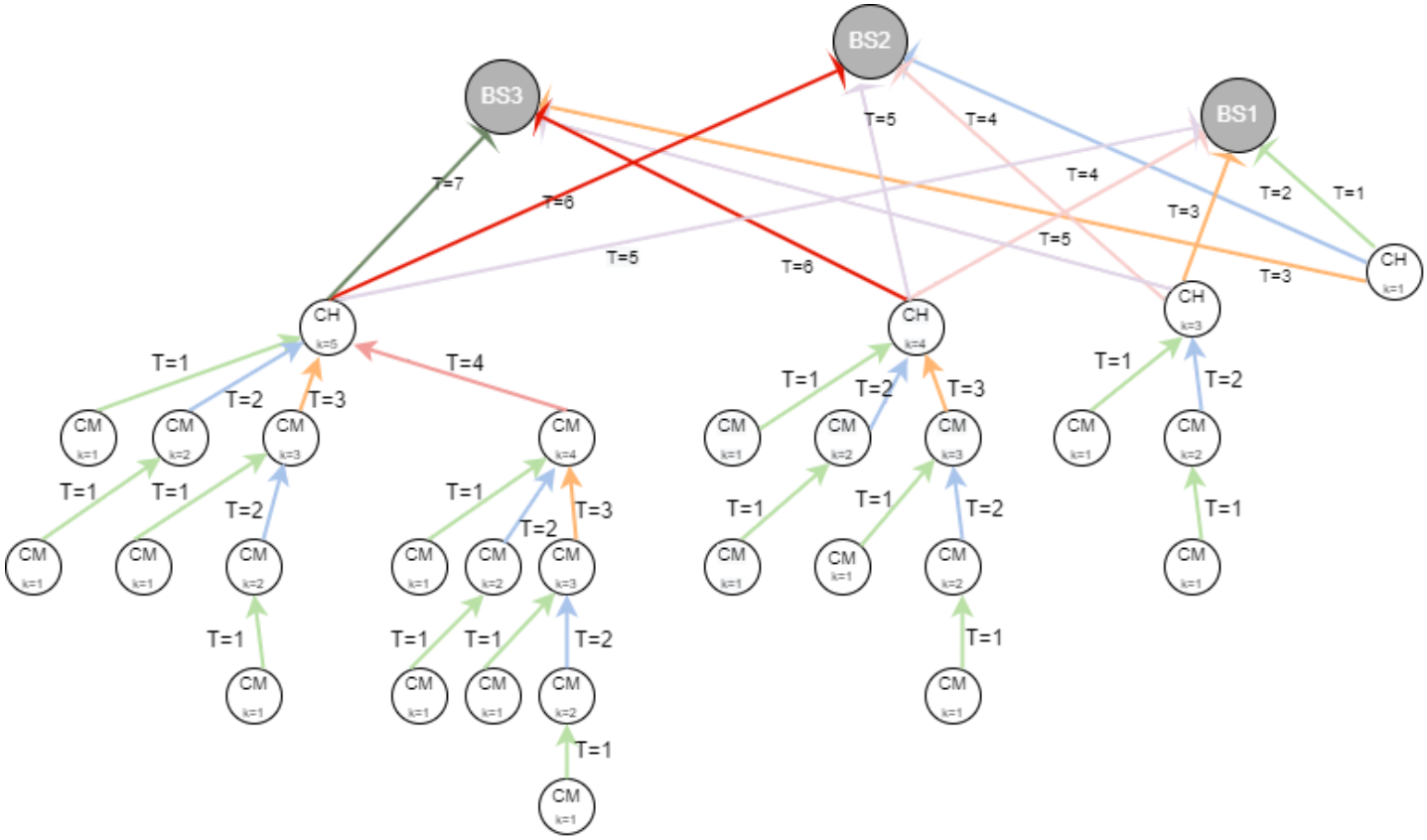


Fig. 3.2. Network size N = 29 for the proposed network structure. The cluster's members are shown as circles with CM. The cluster head is represented by a circle with CH. The base station is represented by a filled circle with BS1,BS2,BS3 . The variable k represents the rank of each node. The existence of a data link is represented by a dashed arrow, and the direction of the arrow indicates the direction of data flow.

As we can see, the total time slots taken by this proposed structure to send all the data from all the nodes of all the clusters to all the base stations in 7 time slots. Parallel data communication is maintained. The structure is also a multi hop structure and the number of hops vary from cluster to cluster depending on its size and kmax values. The next section will prove the algorithms for the proposed design.

Finally, we have to manage how each cluster head will access the base stations. To avoid collisions the proposed structure will arrange the base stations serially such that all the cluster heads will access the base stations in the same order. Which means, by the end of the operations all the base stations will be accessed a similar number of times which will be equal to the number clusters formed in the proposed model. Moreover, in our model all the clusters will be of different sizes so no two cluster heads will finish collecting data from all its nodes at the same time. This means that no two or more clusters will access one particular base station at the same time slot if all base stations are accessed serially in the same order by all the cluster heads. This ensures that there would be no collisions in communication between cluster heads and base stations.

3.5 ALGORITHMS

Let us consider a network of N devices and number of Base stations K. To divide and allocate all the nodes/devices into clusters we use Algorithm 1. The number N is subtracted by the 2^floor(log2(N)) and the resultant number is subtracted from N to find the next cluster size. This process is continued till N becomes zero which means all the nodes are allocated into clusters.

Algorithm 1: (Division of all nodes into clusters)

1. Create an array A to store the Different clusters of each size M=2^p , where p = 1, 2, · · ·.
2. Initialize S=N (total number of nodes)
3. While S is greater than 0
4. Initialize X= 2^(floor(log2(S)))
5. Insert cluster of size X inside the array A

6. Subtract X from M
7. End
8. Return A

Each cluster is internally organized using the DADCNS algorithm as explained in section 3.2 and 3.3. The algorithm for that design is given below.

Algorithm 2: (DADCNS Structure)

1. Inside each cluster, all members will be given a rank. The rank will be an integer value between 1 and p where p=2,3,4….. The rank will be allotted to nodes on the basis of various features such as distance to parent/base station, computing power, location etc.
2. A node with rank k will form k − 1 data links with k − 1 nodes, while these k − 1 nodes. All these k − 1 nodes will become the child nodes of the node with rank k.
3. This higher rank node will become the parent node of the node with rank k.
4. The cluster head is the one with the highest rank in the network and will have k max using equation 1 . The leaf nodes will have rank 1.
5. The Cluster head will connect to the base station.

NOTE: We do not have to explicitly check for collisions as it is impossible to happen in our system if all the base stations are accessed serially in the same order. This is because all the clusters in the network are of different sizes in the form of 2^p where p=1,2,3.. Therefore all cluster heads will take different no of timesots to send aggregated data to each BS. Hence, no two cluster heads will send data to the same base station at the same time slot.

Algorithm 3:- (MAIN ALGORITHM)

1. Initialize the value of |N| and K. Here, |N| denotes the total number of devices in an IoT network and K is the total number of Base Stations.
2. Assign the nodes in multiple DADCNS clusters using Algorithm 1.
3. The nodes within the clusters will be arranged in DADCNS structure using Algorithm 2.
4. Calculate the time slots taken by each cluster to collect and aggregate all the data from all its nodes and forward the data to 1st Base Station using equation 3.
5. Store the time slots in a stack in an ascending order such that the cluster that is able to send the aggregated data to the 1st BS in the least required time slot is on the top and the one which requires the most time slots, in the bottom.
6. Create an array B of Base stations. The index of the array is how the base station is identified i.e. B0 , B1, B2, B3 …. BK. The clusters will access the BS serially in order. This will prevent multiple clusters from accessing the same BS in the same Time Slot.
7. The array of Base Stations will keep count of how many times each Base Station has been traversed by the clusters. All the base stations will be traversed an equal number of times(total number of clusters in the network) at the end.
8. Initialize a variable T which is the number of total time slots passed. Initially T will be 0.
9. Run Loop till all the Base Stations are visited G number of times. G being the total number of clusters in the network.
10. If T == stack.top()

        Increase count in B0

        Then Pop stack.top()

    End.

    For i from 1 to K

        Increase count of B[i] by 1.

        That is, the cluster heads will access subsequent Base stations now.

    End.

    T=T+1

    If all the element in B is same i.e equal to the number of clusters

        Break from Loop;

End.

11. End.
12. Return T(Total Time Slots)
13. Check if the resultant T (total time slot taken) matches with the result of Equation 4.

Let's consider a few examples:

Example 1: Let's consider a network with the number of Nodes N=29 and number of Base stations K=3.

Here the network will be divided into 4 clusters using Algorithm 1. The clusters will be of the size 16, 8, 4 and 1 using Algorithm 1. Each cluster will arrange themselves using Algorithm 2. Using equation 3 we can calculate the time required by each cluster to send aggregated data to the 1st base station. So for clusters of size 16 , 8, 4 and 1 , they are 5, 4, 3 and 1 respectively. Using equation 5 we can calculate the total timeslot required is floor(Log2(29)) + 3 = 7.

Example 2: Let's consider a network with the number of Nodes N=61 and number of Base stations K=7.

Here the network will be divided into 4 clusters using Algorithm 1. The clusters will be of the size 31, 16, 8, 4 and 1 using Algorithm 1. Each cluster will arrange themselves using Algorithm 2. Using equation 3 we can calculate the time required by each cluster to send aggregated data to the 1st base station. So for clusters of size 32, 16 , 8, 4 and 1 , they are 6, 5, 4, 3 and 1 respectively. Using equation 4 we can calculate the total timeslot required is floor(Log2(61)) + 7 = 12.

Example 3: Let's consider a network with the number of Nodes N=36 and number of Base stations K=4.

Here the network will be divided into 2 clusters using Algorithm 1. The clusters will be of the size 32 and 4 using Algorithm 1. Each cluster will arrange themselves using Algorithm 2. Using equation 3 we can calculate the time required by each cluster to send aggregated data to the 1st base station. So for clusters of size 32 and 4 , they are 6 and 3 respectively. Using equation 4 we can calculate the total timeslot required is floor(Log2(36)) + 4 = 9.

Chapter-4 PERFORMANCE ANALYSIS

The proposed model is evaluated and simulated in a custom-built python simulator. The results are then compared to Concurrent Data Collection Network Tree(CDCT) [2] and Time Optimal Concurrent Data Collection Tree(TOCDCT) [4]. These models are explained in chapter 2. The simulated results are then compared to see if the proposed model is actually faster than the two mentioned or not. The following sections will discuss the performance analysis further.

4.1 ALGORITHM ANALYSIS

Let's take an Example And analyze algorithms mentioned in section 3.4 accordingly:

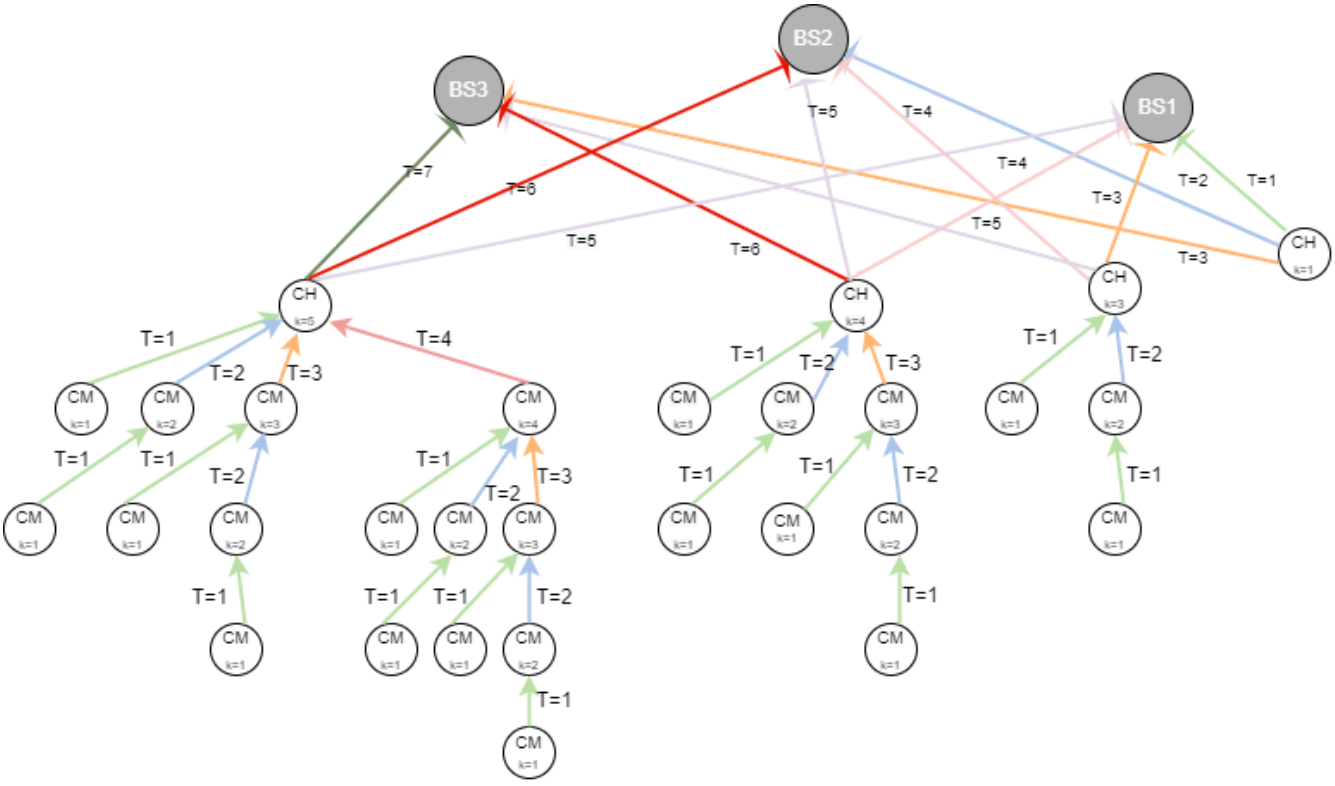


Fig. 4.1. Network size N = 29 for the proposed network structure. The cluster's members are shown as circles with CM. The cluster head is represented by a circle with CH. The base station is represented by a

filled circle with BS1,BS2,BS3 . The variable k represents the rank of each node. The existence of a data link is represented by a dashed arrow, and the direction of the arrow indicates the direction of data flow.

Consider a network with |N| = 29 and K = 3.(FIG 4.1)

1. The nodes will be divided into clusters of [16,8,4,1]
2. Each cluster will be internally arranged according to the DADCNS algo.
3. Then we calculate the kmax of each cluster head (which is equal to the timeslot in which the respective cluster head can forward collected data to the first BS).
4. Push the calculated values into the stack S.

   Stack S > TOP **[1 3 4 5]** BOTTOM
5. Create an array A for the count of times each base station is visited.

   Array A> **[0 0 0]**

   BS1 BS2 BS3
6. Run First Loop till the stack S is empty.
   a. time_slot = 1

      S.top() == time_slot

      S.pop()

      // S > [3 4 5]

      A[0]+=1

      //A > [1 0 0]
   b. time_slot = 2

      S.top() != time_slot

      A[1]+=1

      //A > [1 1 0]
   c. time_slot = 3

      S.top() == time_slot

      S.pop()

      //S>[4 5]

      A[0]+=1

A[2]+=1

//A > [2 1 1]

d. time_slot = 4

S.top() == time_slot

S.pop()

//S> [5]

A[0]+=1

A[1]+=1

A[2]+=1

//A > [3 2 2]

e. time_slot = 5

S.top() == time_slot

S.pop()

//S> [ ]

A[0]+=1

A[1]+=1

A[2]+=1

//A > [4 3 3]

f. Stack S is empty. END LOOP.

7. Run Second Loop till Last cluster (kmax =5) has accessed all remaining BSs.

a. time_slot = 6

A[1]+=1

//A > [4 4 3]

b. time_slot = 7

A[2]+=1

//A > [4 4 4]

c. All the Base Stations are accessed an equal number of times (equal to number of clusters). EXIT LOOP.

8. Return time_slot

The resultant time slot is found to be 7 which complies with the result from equation 4. The above mentioned example is tested on the model proposed in CDCT[2] and TOCDCT[4] and the results are found to be 8 and 8 respectively. So our model is performing the operation in lesser time slots.

4.2 SIMULATION RESULTS

The simulations were performed in a python and c++ simulator that was created. The inputs of Nodes and base stations were varied and the results were then observed. The number of nodes (N) were increased from 30 to 300 in a random fashion so that the inputs do not have any correlation with each other. Similarly the number of Base stations(K) were also increased from 1 to 10. The results were compared against the CDCT and TOCDCT model. The test was divided into two formats.

Since there are two factors, namely number of Nodes(N) and Base Stations(K) that are affecting our final result (Timeslots T), two cases were taken. Case I would have a gradual increase of the number of Nodes from 30 to 300 while keeping the number base stations(K) constant. Case II will have an increase of Base Station(K) from 1 to 10 while keeping the number nodes constant. We took three different scenarios in each case. The results are displayed in both table and bar chart format.

4.2.1 CASE I : Keeping the number of Base Stations(K) constant

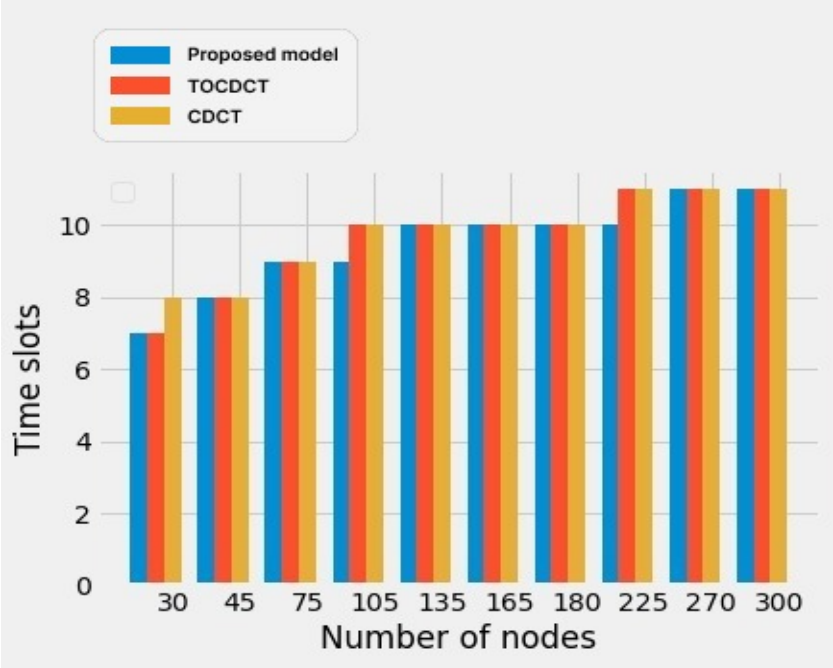
I. Scenario 1:

Let's consider a case where the number of Base Stations K = 3 is kept constant and the number for nodes is increased from 30 to 300. The time slots of CDCT, TO CDCT and our proposed model are compared.

TABLE 4.1

The number of Nodes increased from 30 to 300 while keeping the number base stations(K) constant at K=3.

Number of Nodes (N)	Number of Base Stations(K)	Time slots(T) CDCT	Time slots(T) TO-CDCT	Time slots(T) Proposed Model
30	**3**	**8**	**7**	**7**
45	**3**	**8**	**8**	**8**
75	**3**	**9**	**9**	**9**
105	**3**	**10**	**10**	**9**
135	**3**	**10**	**10**	**10**
165	**3**	**10**	**10**	**10**
180	**3**	**10**	**10**	**10**
225	**3**	**11**	**11**	**10**
270	**3**	**11**	**11**	**11**
300	**3**	**11**	**11**	**11**



Graph 4.1 The number of Nodes increased from 30 to 300 while keeping the number base stations(K) constant at K=3.

Observations:

As observed from Graph 4.1, the timeslots of all three models are very much similar for K=3. Although for N=30, our proposed model and Time Optimal CDCT performs better than CDCT. For N=105 and N=225 our model performs slightly better than both CDCT And Time Optimal CDCT. So for a smaller number of base stations, all three model perform very much equally with our proposed model being just a little bit faster.

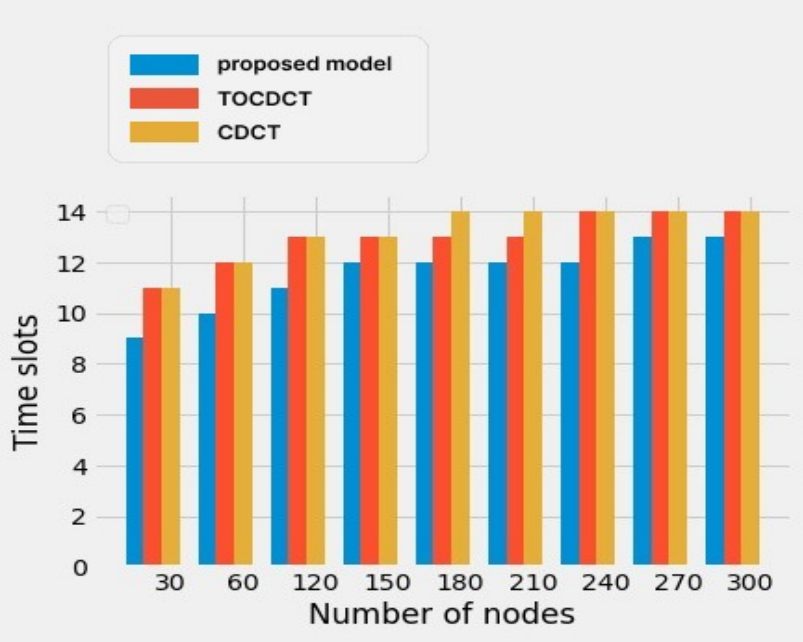
II. Scenario 2:

Let's consider a case where the number of Base Stations K = 5 is kept constant and the number for nodes is increased from 30 to 300. The time slots of CDCT, TOCDCT and our proposed model are compared.

TABLE 4.2

The number of Nodes increased from 30 to 300 while keeping the number base stations(K) constant at K=5.

Number of Nodes (N)	Number of Base Stations(K)	Time slots(T) CDCT	Time slots(T) TO-CDCT	Time slots(T) Proposed Model
30	**5**	**11**	**11**	**9**
60	**5**	**12**	**12**	**10**
120	**5**	**13**	**13**	**11**
150	**5**	**13**	**13**	**12**
180	**5**	**14**	**13**	**12**
210	**5**	**14**	**13**	**12**
240	**5**	**14**	**14**	**12**
270	**5**	**14**	**14**	**13**
300	**5**	**14**	**14**	**13**



Graph 4.2 The number of Nodes increased from 30 to 300 while keeping the number base stations(K) constant at K=5.

Observations:

As observed from Graph 4.2, the timeslots of our proposed model seem to be better than CDCT and TOCDCT. Our model performed better in all the cases. For low and high numbers of Nodes, CDCT and TO CDCT seem to provide similar results, while our proposed model being faster in all the test cases.

III. Scenario 3:

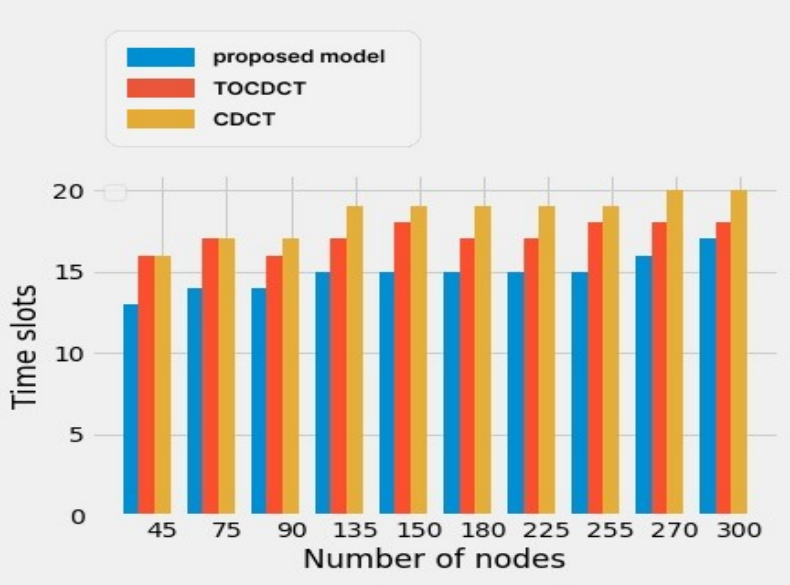
Let's consider a case where the number of Base Stations K = 8 is kept constant and the number for nodes is increased from 30 to 300. The time slots of CDCT, TOCDCT and our proposed model are compared.

TABLE 4.3

The number of Nodes increased from 30 to 300 while keeping the number base stations(K) constant at K=8.

Number of Nodes (N)	Number of Base Stations(K)	Time slots(T) CDCT	Time slots(T) TO-CDCT	Time slots(T) Proposed Model
45	**8**	**16**	**16**	**13**
75	**8**	**17**	**17**	**14**
90	**8**	**17**	**16**	**14**
135	**8**	**19**	**17**	**15**
150	**8**	**19**	**18**	**15**
180	**8**	**19**	**17**	**15**
225	**8**	**19**	**17**	**15**
255	**8**	**19**	**18**	**15**
270	**8**	**20**	**18**	**16**

300	**8**	**20**	**18**	**16**



Graph 4.3 The number of Nodes increased from 30 to 300 while keeping the number base stations(K) constant at K=8.

Observations:

As observed from Graph 4.3, the timeslots of our proposed model seem to be better than CDCT and TOCDCT. Our model performed better in all the cases. For a low number of Nodes, CDCT and TO CDCT seem to provide similar results, while our proposed model being faster in all the test cases. For a higher number of nodes both our model and TO CDCT performs better than CDCT. We can conclude that as the number of Nodes get higher our models also perform better than CDCT and TO CDCT for the same number of base stations.

4.2.1 CASE I : Keeping the number of Nodes(N) constant

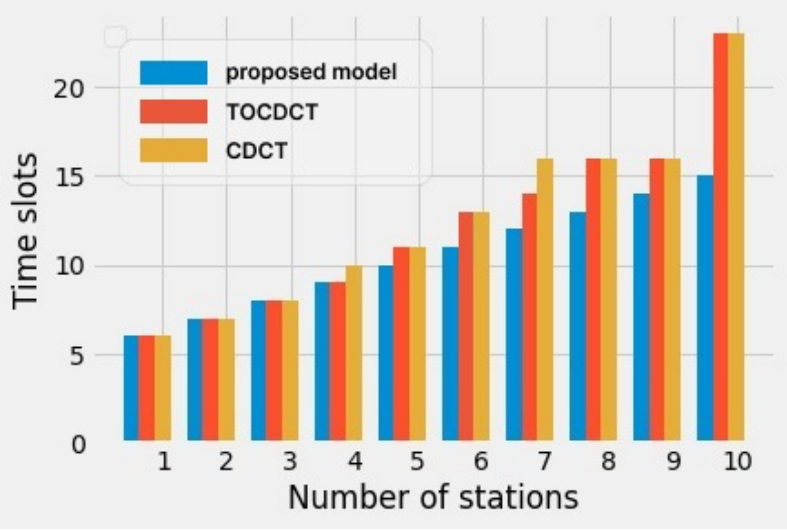
I. Scenario 1:

Let's consider a case where the number of Nodes N = 45 is kept constant and the number for base stations K is increased from 1 to 10. The resulting time slots of CDCT, TO CDCT and our proposed model are compared.

TABLE 4.4

The number of Base Station(K) increased from 1 to 10 while keeping the number of Nodes(N) constant at N=45.

Number of Nodes (N)	Number of Base Stations(K)	Time slots(T) CDCT	Time slots(T) TO-CDCT	Time slots(T) Proposed Model
45	**1**	**6**	**6**	**6**
45	**2**	**7**	**7**	**7**
45	**3**	**8**	**8**	**8**
45	**4**	**10**	**9**	**9**
45	**5**	**11**	**11**	**10**
45	**6**	**13**	**13**	**11**
45	**7**	**16**	**14**	**12**
45	**8**	**16**	**16**	**13**
45	**9**	**16**	**16**	**14**
45	**10**	**23**	**23**	**15**



Graph 4.4 The number of Base Station(K) increased from 1 to 10 while keeping the number of Nodes(N) constant at N=45.

Observations:

As observed from Graph 4.4, the timeslots of the proposed model seem to be better than CDCT and TOCDCT. Our model performed better in all the cases. For a low number of Base Stations, our model, CDCT and TO CDCT seem to provide similar results, For a higher number TO CDCT performs better than CDCT while the proposed model performs better in all the cases. The interesting case is when K=10. The proposed model seems to perform much better than CDCT and TO CDCT. This might be due to the case that the proposed model structure is better suited to the number of Nodes and Base stations in this case while the ring structures of CDCT and TO CDCT can converge the data between all nodes and base stations as quickly.

II. Scenario 2:

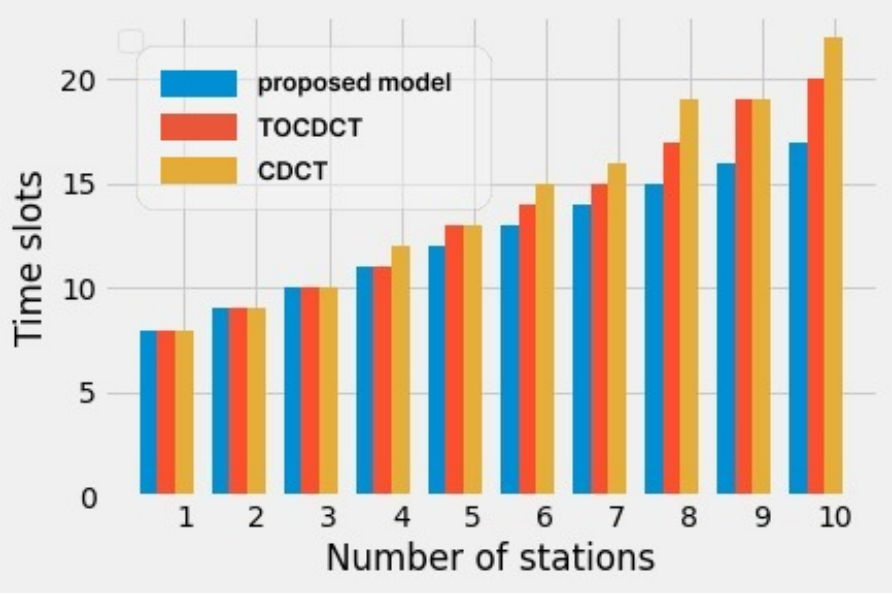
Let's consider a case where the number of Nodes N = 135 is kept constant and the number for base stations K is increased from 1 to 10. The resulting time slots of CDCT, TO CDCT and our proposed model are compared.

TABLE 4.5

The number of Base Station(K) increased from 1 to 10 while keeping the number of Nodes(N) constant at N=135.

Number of Nodes (N)	Number of Base Stations(K)	Time slots(T) CDCT	Time slots(T) TO-CDCT	Time slots(T) Proposed Model
135	**1**	**8**	**8**	**8**
135	**2**	**9**	**9**	**9**
135	**3**	**10**	**10**	**10**
135	**4**	**12**	**11**	**11**
135	**5**	**13**	**13**	**12**

135	**6**	**15**	**14**	**13**
135	**7**	**16**	**15**	**14**
135	**8**	**19**	**17**	**15**
135	**9**	**19**	**19**	**16**
135	**10**	**22**	**20**	**17**



Graph 4.5 The number of Base Station(K) increased from 1 to 10 while keeping the number of Nodes(N) constant at N=135.

Observations:

As observed from Graph 4.5, the timeslots of the proposed model seem to be better than CDCT and TOCDCT. Our model performed better in all the cases. For a low number of Base Stations, our model, CDCT and TO CDCT seem to provide similar results, For a higher number TO CDCT performs better than CDCT while the proposed model performs better in all the cases. We can see that the proposed model's performance gets better and better as the number of base stations K increases.

III. Scenario 3:

Let's consider a case where the number of Nodes N = 255 is kept constant and the number for base stations K is increased from 1 to 10. The resulting time slots of CDCT, TO CDCT and our proposed model are compared.

TABLE 4.6

The number of Base Station(K) increased from 1 to 10 while keeping the number of Nodes(N) constant at N=255.

Number of Nodes (N)	Number of Base Stations(K)	Time slots(T) CDCT	Time slots(T) TO-CDCT	Time slots(T) Proposed Model
255	**1**	**8**	**8**	**8**
255	**2**	**9**	**9**	**9**
255	**3**	**11**	**11**	**10**
255	**4**	**12**	**12**	**11**
255	**5**	**14**	**14**	**12**
255	**6**	**16**	**15**	**13**
255	**7**	**18**	**17**	**14**
255	**8**	**19**	**18**	**15**
255	**9**	**22**	**19**	**16**
255	**10**	**23**	**21**	**17**

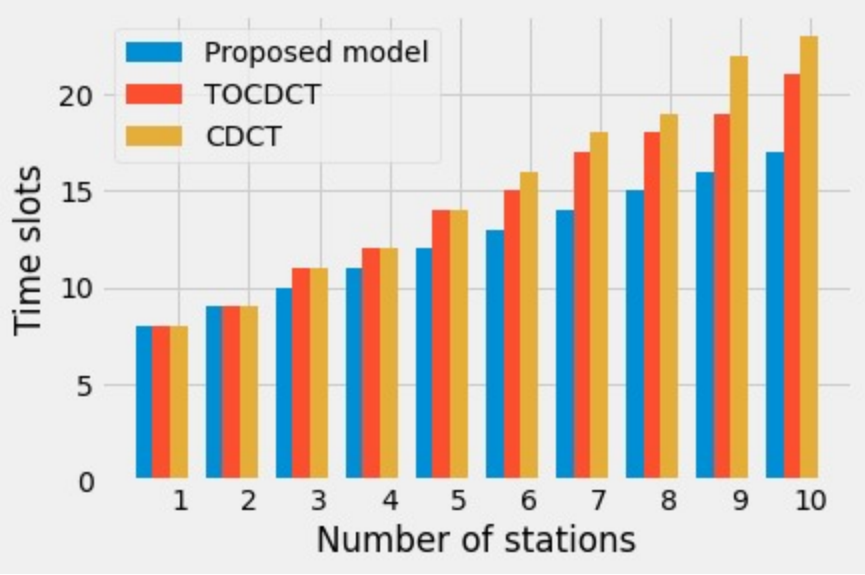


Fig Graph 4.6 The number of Base Station(K) increased from 1 to 10 while keeping the number of Nodes(N) constant at N=255.

Observations:

As observed from Graph 4.6, the timeslots of the proposed model seem to be better than CDCT and TOCDCT. Our model performed better in all the cases. Similar to scenario 2, for a low number of Base Stations, our model, CDCT and TO CDCT seem to provide similar results, For a higher number TO CDCT performs much better than CDCT while the proposed model performs better in all the cases. We can see that the proposed model's performance gets better and better as the number of base stations K increases.

4.3 RESULT DISCUSSION

From the experiments performed in section 4.2, it is observed that for all cases the proposed model performs better than both Concurrent Data Collection Tree(CDCT) and Time Optimal CDCT. It is also seen that as the number of nodes (N) and base stations(K) are increased, the resulting timeslots from the proposed network structure performs even better compared to the other two models.

The simulation results support the findings of our analysis, which were covered in the previous section. When compared to networks using CDCT and Time Optimal CDCT, the total number of time-slots needed by the network using the proposed time optimal data collection trees is significantly lower.

For instance, the duration of data collection in CDCT is 17 time slots and Time Optimal CDCT is 16 time slots when |N| = 150 and k = 7, whereas the duration of data collection in the proposed approach is 14 time slots for the same values of |N| and k.

Similar to this, the duration of data collection in CDCT and Time Optimal CDCT is 23 timeslots and 20 timeslots respectively when |N| = 250 and k = 10, while the duration of data collection in the proposed approach is 17 timeslots for the same values of |N| and k.
The gains of the suggested network structures increase as |N| and K are increased, and it is demonstrated that increase in clusters causes the number of time slots T to increase slowly. The number of clusters will rise proportionally as |N| increases, and as more devices participate in concurrent transmissions, the proposed tree structure's total number of slots T will be lower than that of CDCT and Time Optimal CDCT.

Chapter-5 CONCLUSIONS

5.1 CONCLUSIONS

In this study, we propose an ideal method for collecting data concurrently in IoT systems. A growing number of Internet of Things (IoT) applications rely on such shared systems for their data needs due to the enhanced benefits of shared device infrastructure. Concurrent data delivery is required in order to maintain the freshness of data as multiple such applications request data at once. Furthermore, in real-time systems, the timely delivery of data is essential for making critical decisions. Here, a combination of network topologies is used in the proposed network structure to reduce the delays. The results of the simulation and performance analysis demonstrate that the proposed process outperforms the currently used data collection methods, CDCT and Time Optimal CDCT.

It is predicted that public and private internet of things (IoT) systems will soon be connected to create an IoT federation. IoT devices will be shared among various parties under these connected systems. On the same collection of IoT devices, various data collection processes started by various users can run concurrently.

Specifically created for concurrent data collection processes in IoT systems. The suggested network structure can reduce the delays caused by multiple data collection processes running at once. According to the findings of this project, the suggested idea can result in shorter data collection times than a network structure that is currently in use and was created for a single data collection process. Also provided are comprehensive instructions for obtaining workable transmission schedules for the suggested network structure.

Network formation can be implemented in a centralized or decentralized way to accommodate various applications. For networks of various sizes, two network formation strategies are developed to deliver optimal results. A multiple-cluster 2-hop network structure, a single-chain network structure, a minimum spanning tree network structure, and a collection tree network

structure are used to evaluate the performance of the suggested network structure. Among the network structures mentioned above, the proposed network structure is demonstrated to be the most effective in terms of data collection time. The suggested network structure can significantly cut down on data collection time while maintaining acceptable levels of overall communication distance and network lifetime.

5.2 FUTURE SCOPE

The Internet of Things has become a dominant technology globally. In a short period of time, it has become extremely popular. In addition, the automation of IoT devices has become simple thanks to advancements in artificial intelligence and machine learning. In essence, IoT devices are combined with AI and ML programmes to properly automate them. As a result, the IoT has broadened the range of industries in which it can be applied. We'll talk about the uses and potential of IoT in the healthcare, automotive, and agricultural industries in this section.

The method for obtaining the transmission schedules in the suggested network structure can be easily changed to take other optimization constraints or criteria into account. The total communication distance of the data collection tree is a common worry for mobile networks because it may negatively impact the battery life of mobile devices. With the aid of clustering algorithms with predetermined cluster sizes, N2N communication distance between each member of a cluster of the proposed structure can be minimized. By using travelling salesman problem solvers to change the order of the nodes inside each loop, it is possible to further reduce this parameter and shorten the length of the ring's overall path. To turn the procedures into a multi-objective optimization process, additional criteria, like channel quality and bandwidth, can also be included. Interferences caused by concurrent transmissions are a different issue, but they can be avoided or reduced by imposing minimum separation requirements on conflicting nodes when creating workable transmission schedules. Additionally, using different communication

channels, which is a practical option for the majority of contemporary transceiver modules, can reduce interferences among IoT devices.

Fast data collection from IoT sensors can help receive data in lesser delay form ever before. which can eventually help benefit globally since IoT has become a day to day application in our lives. This project could be extended to various domains. For this particular research, we have only looked at the structure of the network between the nodes and the base stations. In future we can also look into the frequency of each device and the energy consumption of these data collection processes and try to minimize it. This could in turn help us to reduce the use of energy to operate an IoT network.

By processing big data, cloud computing is currently generating a lot of interest across many industries. where information is gathered from a variety of sources, including sensor networks, social networks, and automobiles. Concerns about the security of the data transferred to the cloud data center from the aforementioned sources can still be addressed. To support the data collection from sensors to the cloud, a common architecture is required.

Any communication system that adheres to a specific Quality of Service (QoS) for each communication application is built on a foundation of network protocols. Remote system architecture becomes increasingly capable as installed technology advances quickly, and its topological structure and correspondence become more unpredictable. The proposed model could be extended to other data collection domains as well including data collection from WSNs etc.

5.3 APPLICATIONS

Everyone wants a job done for them without any effort as the world's population shifts toward relying more on technology than manual methods. Internet of Things essentially refers to computing devices that transmit and receive data over the internet. Considering its advantages

and the degree of comfort people are experiencing, IoT is becoming a significant part of their lives. IoT can assist humanity in many ways, some of which are listed below:

1. The medical industry: Extensive implementation is possible. Checkups, wearable health devices, telemedicine, and many other things.

2. Smart Homes: Several devices, including Google Home, Amazon's Alexa, and Nest, have been introduced. Each of these devices has a specific function that makes it possible for members of a household to communicate with one another online and improves the quality of our lives.

3. Smart Cities: Time-wasting traffic jams are the major issue in large cities, but IoT is facilitating connectivity and information sharing to allow for proactive situational management. security systems, cutting-edge parking systems.

There are numerous additional industries, including manufacturing, advanced power supply, planning, industrial automation, and the digitization of cities in developing nations (see, for instance, Mark Zuckerberg's JARVIS). There are countless opportunities.

References

[1] C.-T. Cheng, K. T. Chi, and F. C. Lau, “A delay-aware data collection network structure for wireless sensor networks,” IEEE sensors journal, vol. 11, no. 3, pp. 699–710, Mar. 2011.

[2] C. 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, Apr. 2017.

[3] Ozlem D Incel, Amitabha Ghosh, and Bhaskar Krishnamachari, “Fast Data Collection in Tree-Based Wireless Sensor Networks” IEEE Transactions on Mobile Computing 11(1),February 2012

[4] Arvind Kumar, Rakesh Matam, and Mithun Mukherjee, “Time Optimal Concurrent Data collection Trees for IoT Applications” IEEE International Systems Conference(SysCon), April 2021

[5] J. N. Al-karaki and A. E. Kamal, “Routing techniques in wireless sensor networks: a survey,” IEEE Wireless Communications Mag., vol. 11, no. 6, pp. 6–28, December 2004.

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

[7] S. Lindsey and C. S. Raghavendra, “PEGASIS: Power-efficient gathering in sensor information systems,” in Proc. IEEE Conf. Aerospace, vol. 3, Big Sky, Montana, USA, March 2002, pp. 1125–1130.

[8] H. O. Tan and ¨ ´I. Korpeo ¨ glu, “Power efficient data gathering and ag- ˆ gregation in wireless sensor networks,” ACM SIGMOD Record, vol. 32, no. 4, pp. 66–71, December 2003.

[9] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, “The collection tree protocol,” TinyOS Enhancement Proposals (TEP) 123, December 2007.

[10] C. Florens, M. Franceschetti, and R. J. McEliece, “Lower bounds on data collection time in sensory networks,” IEEE Jour. Selected Areas in Communications, vol. 22, no. 6, pp. 1110–1120, August 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. of the 12th annual International Conf. on Mobile computing and networking. ACM, 2006, pp. 262–273.

[12] I. Solis and K. Obraczka, "The impact of timing in data aggregation for sensor networks," in IEEE International Conf. on Communications (IEEE Cat. No. 04CH37577), vol. 6, 2004, pp. 3640–3645.

[13] M. Song and B. He, "Capacity analysis for flat and clustered wireless sensor networks," in Proc. Int. Conf. Wireless Algorithms, Systems and Applications, (WASA 2007), Chicago, Illinois, USA, August 2007, pp. 249–253.

[14] V. Annamalai, S.K.S. Gupta, L. Schwiebert, "On tree-based convergecasting in wireless sensor networks", in WCNC '03, pp.

1942–1947.

[15] S. Gandham, Y. Zhang and Q. Huang, "Distributed Time-Optimal Scheduling for Convergecast in Wireless Sensor Networks", Computer Networks, vol. 52, nr. 3, 2008, pp. 610–629.

[16] T. Moscibroda, "The worst-case capacity of wireless sensor networks", in IPSN '07, pp. 1–10

[17] G. Chen, J. Tang, and J. P. Coon, "Optimal routing for multihop socialbased d2d communications in the internet of things," IEEE Internet of Things Journal, vol. 5, no. 3, pp. 1880–1889, 2018.

[18] O. Tekdas, J. H. Lim, A. Terzis, and V. Isler, "Using mobile robots to harvest data from sensor fields," IEEE Wireless Communications Mag., vol. 16, no. 1, pp. 22–28, February 2009

[19] W. Zhao and X. Tang, "Scheduling sensor data collection with dynamic traffic patterns," Parallel and Distributed Systems, IEEE Transactions on, vol. 24, no. 4, pp. 789–802, 2013.

[20] S. Ji, R. Beyah, and Z. Cai, "Snapshot and continuous data collection in probabilistic wireless sensor networks," Mobile Computing, IEEE Transactions on, vol. 13, no. 3, pp. 626–637, 2014.

[21] Z. Cai, S. Ji, J. He, L. Wei, and A. Bourgeois, "Distributed and asynchronous data collection in cognitive radio networks with fairness consideration," Parallel and Distributed Systems, IEEE Transactions on, vol. 25, no. 8, pp. 2020–2029, 2014.

[22] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of MANET and

WSN in IoT urban scenarios," Sensors Journal, IEEE, vol. 13, no. 10, pp. 3558–3567, 2013

[23] P. Maiti, B. Sahoo, A. K. Turuk, and S. Satpathy, "Sensors data collection architecture in the internet of mobile things as a service (iomtaas) platform," in 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE, 2017, pp. 578–582

[24] R. Kolcun, D. Boyle, and J. A. McCann, "Optimal processing node discovery algorithm for distributed computing in IoT," in Proc. 5th International Conf. on the Internet of Things (IoT), 2015, pp. 72–79

APPENDICES

Python 3 simulator used for this project:

```
import math

n=int(input("Enter number of Nodes = "))
k=int(input("Enter number of Base Stations = "))
li=[]
m=n

while m>0:
    x=math.pow(2,math.floor(math.log2(m)))
    li.append(x)
    m-=x
print("The cluster will be of size each = ")

for i in range(len(li)):
    print(li[i],end=" ")
print()

tm=[]
#k=0
for i in range(0,len(li)):
    k=(math.log2(li[i]))+1
    tm.append(k)

print("Timeslots taken by each Cluster Head(CH) to forward aggregated data to the 1st BS")
for i in range(0,len(tm)):
    print(tm[i],end=" ")
```

```
print()

k=int(k)
st=[]
for i in range(len(tm)-1,-1,-1):
    st.append(tm[i])

k=int(k)
arr=[]
print(k)
for i in range(0,k):
    arr.append(0)
print("Intially each base station is visited 0 times as shown: -")

for i in range(0,k):
    print(arr[i],end=" ")

time=0
while len(st)!=0:
    time+=1
    for i in range(1,k):
        if arr[i]<arr[i-1]:
            arr[i]+=1
        else:
            break
    if st[0]==time:
        arr[0]+=1
        st.pop()

for i in range(1,k):
```

```
    arr[i]+=1
    #time+=1
print()
print("All the base stations are visited by all the clusters(whose count is equal to the number of clusters) as shown: -")
for i in range(0,k):
    print(arr[i],end=" ")
print("\nTotal time slots taken = ",time)
```

C++ 14 version:

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    // Write C++ code here
    int N,k;
    cout<<"Enter number of Nodes: ";
    cin>>N;
    cout<<"Enter number of Base Stations: ";
    cin>>k;
    //breaking the nodes into clusters of 2^p where p=1,2,3..
    vector<int> v;
    int M=N;
    while(M>0)
    {
        int x=pow(2,floor(log2(M)));
        v.push_back(x);
        M-=x;
    }
    cout<<"The clusters will be of size each :"<<endl;
```

```
    for(int i=0;i<v.size();i++)
        cout<<v[i]<<" ";
    //calculating time taken by each cluster head to forward aggregated data to the 1st BS
    cout<<endl;
    vector<int> tm;
    for(int i=0;i<v.size();i++)
    {
        int k=log2(v[i])+1;
        tm.push_back(k);
    }
    cout<<"timeslots taken by each Cluster Head(CH) to forward aggregated data to the 1st
BS"<<endl;
    for(int i=0;i<tm.size();i++)
        cout<<tm[i]<<" ";

    stack<int> st;
    for(int i=0;i<tm.size();i++)
    {
        st.push(tm[i]);
    }
    int time=0;
    int arr[k];
    for(int i=0;i<k;i++)
    {
        arr[i]=0;
    }
    cout<<endl;
    cout<<"Intially each base station is visited 0 times as shown: "<<endl;
    for(int i=0;i<k;i++)
    {
        cout<<arr[i]<<" ";
```

```
    }
    while(!st.empty())
    {
        time+=1;
        for(int i=1;i<k;i++)
        {
            if(arr[i]<arr[i-1])
            {
                arr[i]+=1;
            }
            else
                break;
        }
        if(st.top()==time)
        {
            arr[0]+=1;
            st.pop();
        }
    }
    for(int i=1;i<k;i++)
    {
        arr[i]+=1;
        time+=1;
    }
    cout<<endl;
    cout<<"All the base stations are visited by all the clusters(whose count is equal to the number
of clusters) as shown "<<endl;
    for(int i=0;i<k;i++)
    {
        cout<<arr[i]<<" ";
    }
```

```
    cout<<endl;
    cout<<"Total time slots taken: "<<time;
    return 0;
}
```