Dynamic Energy Aware Aggregate Convergecast in a Wireless Sensor Network

by

Muhuji Mshana

A Thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Computer Science

School of Computer Science
Carleton University
Ottawa, Ontario, Canada
September 2012

Copyright ©
2012 - Muhuji Mshana

Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-93629-0

Our file Notre référence

ISBN: 978-0-494-93629-0

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Wireless Sensor Network (WSN) is a network of sensors deployed or set up in an environment where sensors collect specified data and forward them to a sink for processing. The process of sending packets from sensor nodes towards the sink node from all the sensor nodes is known as Convergecast. Convergecast problem in WSN is how to successfully send data from the sensor nodes to the sink in a short time.

We introduce in this work an energy efficient, collision aware and dynamic convergecast solution for a tiled WSN. We introduce Dynamic Dominating Set (DDS) algorithm which uses dynamic dominating set to form a network graph based on two metrics: neighbor count and energy of each sensor. Dynamic nature of the network results from changes in metrics when a network tree is constructed due to decrease in sensors energy every time they transmit. We introduce Local Collision Aware Convergecast (LCAC) algorithm which uses dynamic tree graph from DDS algorithm and a local collision model to forward messages to the root node (sink) using aggregate convergecast.

We proved that DDS algorithm takes $O(N^2)$ time to construct a tree in one tile where N is the number of nodes on the network. We also proved that LCAC algorithm takes $O(N)$ time to converge all data from sensors to sink. We simulated our algorithms in Netlogo. Our test results proved to have minimal execution time, are energy efficient and reduce collision on the network.

Table of Contents

Abstract	ii
Table of Contents	iii
List of Tables	vi
List of Figures	vii
1 Introduction	1
1.1 Motivation and Problem Statement	3
1.2 Assumptions	6
1.3 Summary of Contributions	7
1.4 Organization of Thesis	8
2 Background	9
2.1 Convergecast in WSN	9
2.1.1 Types of Convergecast	10
2.1.2 Properties of Convergecast	11
2.1.3 Why using Convergecast	12
2.2 Classification Approach and Methodology	13
2.2.1 Design Objectives	13
2.2.2 Design Constraints and Assumptions	18

3	Literature Review	22
3.1	Minimizing Schedule Length	22
3.2	Minimizing Latency	37
3.3	Minimizing Energy Consumption	38
3.4	Mobility and Multi-path Support	40
4	Dynamic Dominating Set Algorithm	42
4.1	Network Model	42
4.2	Dynamic Dominating Set	44
4.2.1	Dominating Set	44
4.2.2	Dynamic Dominating Set	44
4.3	Dynamic Dominating Set Algorithm	45
4.4	Theoretical Analysis of DDS Algorithm	47
5	Local Collision Aware Convergecast Algorithm	50
5.1	Convergecast	50
5.2	Local Collision Model	51
5.3	LCAC Algorithm	53
5.4	System Flow	55
5.5	Theoretical Analysis of LCAC Algorithm	55
6	Analysis of Algorithms	60
6.1	Theoretical Analysis	60
6.2	Simulations	62
6.2.1	Simulation Setup	62
6.2.2	Performance Metrics	67
6.2.3	Initial Test Results	68
6.2.4	Test Results with TDMA	74

6.3	Performance Evaluation	87
6.3.1	Time of Execution	87
6.3.2	Energy Conservation	88
6.3.3	Collision Avoidance and Minimization	89
6.3.4	Mass Network Coverage	90
7	Conclusion and Future Work	91
7.1	Future Work	92
	List of References	95

List of Tables

1	Parameter settings for initial test cases	69
2	Parameter settings for Node density tests	75
3	Parameter settings for Node degree tests	80

List of Figures

1	An example of a Wireless Sensor network	2
2	Convergecast vs. Broadcast: (a) Shows Convergecast flow to sink (b) Shows broadcast flow from Sink	10
3	Mobile sink using TSP to navigate to all sensor nodes in a shorter route by getting in the radio range as shown as a UDG	29
4	Tile Network plane with nodes forming a tree in every tile. Tile sinks form a secondary network to connect to the Sink.	43
5	Node A receives transmissions from node B and C in the same time slot.	52
6	Node A hears transmission from B while it is receiving a transmission from C.	53
7	Snapshot of WSN model developed in Netlogo	62
8	WSN model's control panel	64
9	Netlogo's world showing a blue node at the center with its transmission zone in Grey for values of Neighbor-degree-distance at: (a) 2 (b) 3 (c) 4 (d) 5 (e) 6	66
10	A snapshot of a running model showing how nodes take different roles on every cycle.	67
11	Low density test- Graphs of (a) Average residual energy (b) Average nodes alive and (c) average Number of nodes transmitted with time .	70

12	Medium density test:- Graphs of (a) Average residual energy (b) Average nodes alive and (c) average Number of nodes transmitted with time	71
13	High density test:- Graphs of (a) Average residual energy (b) Average nodes alive and (c) average Number of nodes transmitted with time .	72
14	Node density Test 1- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	76
15	Node density Test 2- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	77
16	Node density Test 3- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	78
17	Node density Test 4- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	79
18	Node degree Test 1- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	81
19	Node degree Test 2- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	82
20	Node degree Test 3- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	83
21	Node degree Test 4- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	84
22	Node degree Test 5- Graph of (a) Average Residual energy (b) Average nodes alive (c)Average transmitting nodes and (d) Nodes on collision	85
23	Graph of time (ms) vs. Number of nodes	86
24	Graph of time (ms) vs. Neighbor degree radius (unit)	87

Chapter 1

Introduction

Wireless networks are being well accepted in the technological world where most aspects of computer and information systems tend to converge to the wireless environment. Wireless network can simply be defined as a type of network where nodes or devices exchange information through wireless medium [1]. The medium used is radio waves where different radio frequency ranges are defined to be different wireless network standards.

In computer networks there are two types of wireless networks; infrastructure network and ad hoc network. In infrastructure network, nodes in the network are connected to a central node which issues address and controls communication. Like most home or office networks, computers are connected to an Access Point (AP) or a wireless router. Wireless ad hoc network has no central node for all nodes to connect to and receive instruction from. Each node in a network is a master and slave of other nodes.

Wireless Sensor Network (WSN) is a network of sensors deployed or set up in an environment where sensors collect specified data and send them to a sink for processing and analysis [1]. A sink can be a server, router, base station or any device that can act as a collector of data. WSN is an ad hoc network where each node has to know its neighbors and know which one to relay its packets to so as to convey

its data to the server. Most of these sensors are designed to be deployed in remote environments and stay there until they serve the purpose of the network or run out of power. For sensors to have longer life time in the network, they are made with limited processing capabilities, memory, mobility, communication means (transceiver) and maximized energy storage capacity. The environment being monitored also affects the size and structure of sensor. Thus over time researchers have proposed and come up with different protocols, designs and strategies to ensure efficient and thorough ways of data collection in WSNs.

Figure 1: An example of a Wireless Sensor network

Examples of WSN are sensors systems deployed in a building for temperature control, Temperature and radioactive leakage systems in Nuclear power plants, Hydro and nutrient Monitoring systems in irrigation and agriculture and Spy bots or surveillance Systems researched and used by Military forces. As seen in different

WSNs, urgency of information from the sensors to sink differs in different networks. Radioactive leakage and temperature control systems require live data from sensors and also a location included in the data to further narrow down where exactly a leakage is. While a network like Surveillance system in the bank vault or temperature control system can be monitored with even a delayed routing of its data from sensors. Different pattern and schedule of data collection brings different output, delays, latency and energy usage in the network.

1.1 Motivation and Problem Statement

The idea of using a tiled network was inspired by the work in [2] where they used a layered network to study network efficiency and latency in WSN convergecasting by using Maximum Independent Sets (MIS). Dividing a network into sub networks proved to be more energy efficient and time saving due to parallel operations in sub networks. We are using a tiled network to model a convergecast schedule where data sensed is coming from nodes in the tile. By using Connected Dominating Sets (CDS) the nodes relay the sensed information from the sensing nodes via intermediate tile sinks to the network sink. Nodes with energy above a specified energy threshold are selected to be dominants in each tile network and a tile sink is picked from a set of dominants which has the highest energy and neighbor count. With varying energy consumption in sensor nodes, different nodes will be assuming the role of dominants and tile sink; thus the dynamic nature of our algorithm.

The use of aggregation in convergecast was seen to be efficient in [2–5] as transceiver activities were minimized. Nodes saved energy and the network gained more life time. Our algorithm will use aggregated convergecast technique as a strategy to save energy and prolong network lifetime.

To further cut down the use of energy in transmission within nodes in the system;

1. We use connected dominating sets as our key message forwarders. Each tile will have a network tree with dominant nodes as parent nodes and a tile sink as a root node.
2. Each sensing node is connected to a dominating node within its transmission range.
3. After convergecast in tile network is complete, each tile sink acts as node with information which it needs to send to the network sink. So a second layer connected dominating set is formed to forward all the messages to the sink.

Transmission over wireless environment has high collision rates because one channel can be used by more than one pair of nodes. Many algorithms used TDMA/CDMA as a collision reduction mechanism. We looked for different collision reduction mechanism by checking for collision at local neighborhood of nodes.

As will be seen in the next chapter, convergecast is a data collection methodology in WSN. Like any other data collection methodologies, convergecast schedules are supposed to be fast, energy efficient and have high quality of service. Different approaches and ideas can be used to implement convergecast algorithms that obey all properties of convergecast in WSN. The problem is ensuring the algorithms are able to run in minimal times so less sensor resources are used. Hence the convergecast problem is *finding a convergecast schedule for collecting data in the network which runs in minimal time.*

Mathematically, convergecast problem can be defined as follows: In a network $G = (V, E)$ with N nodes where V is a set of nodes including sink node v_0 and E is a set of connections between nodes. Assume all nodes in $V' = \{v_1(m_1), v_2(m_2) \dots v_N(m_N)\}$ have a message m_i to be sent to sink node v_0 . Convergecast problem is finding a data collection schedule τ which collects all messages from sensor nodes $\{v_1(m_1), v_2(m_2) \dots v_N(m_N)\}$ to a sink node v_0 in minimal time T_{min} . So at the end of

the schedule all messages are collected to the sink $v_0(\{m_1, \dots, m_N\})$. The following terms will be used to define convergecast problem.

Definition 1. *Schedule τ is a set of edges between nodes in order of how nodes forward messages to the sink. τ can be presented as $\{(u_1, v_1)_{t_1}, (u_2, v_2)_{t_2}, \dots (u_i, v_i)_{t_i}, \dots\}$ where u_i denotes a transmitting node and v_i denotes a receiving node in different time units $\{t_1, t_2, \dots, t_i, \dots\}$, respectively.*

Definition 2. *Minimal time T_{min} of a schedule is the minimum time a convergecast schedule τ uses to collect data from all sensor nodes to sink.*

Assume nodes $u \in V$ and $v \in V$ are sensor nodes in the network which are within each other's transmission range. Let $e_{uv} \in E$ denote the connection between the two nodes. Node u is farther from the sink node r compared to v , hence node u will use less energy to forward its message to v than sending it straight to the sink. Let $t_{u,v}$ denote the time used by node u to send a packet to node v . If these nodes send their packets by obeying a collision free transmission then the convergecast problem is finding a minimum running time for the schedule τ :

$$T_{min}(\tau) = \min_{\tau} T(\tau)$$

where;

$$\tau = \{(u_1, v_1)_{t_1}, (u_2, v_2)_{t_2}, \dots (u_i, v_i)_{t_i}, \dots\}$$

and

$$T(\tau) = \sum t_{u_i, v_i}$$

Schedule time minimization is the main problem in many algorithms in this area of research. Convergecast problem can also be defined as an energy optimization problem in WSN. Nodes on the network use energy every time they transmit or

receive messages to a point when they run out of energy. We refer to the energy available on the node's battery at any time as **residual energy**.

We address the convergecast problem by proposing a Dynamic Dominating Set (DDS) algorithm that constructs a network tree using a dominating set based on nodes with maximum residual energy and highest neighbor degree. We also introduce Local Collision Aware Convergecast (LCAC) algorithm. LCAC is an aggregate convergecast algorithm proposed in this solution. LCAC runs on a network tree constructed by DDS, it takes into account local neighbor state of each node taking part in convergecast. Each node scans its neighborhood and parent's child set before it transmits.

Using DDS and LCAC algorithms in a WSN network not only minimizes convergecast schedule time but also optimizes efficient energy consumption of the network. DDS implements a load balancing scheme between network nodes by its dynamic tree construction over the network. LCAC reduces transmission collisions between nodes with its local collision awareness model. Section 1.2 will give assumptions made in our model of WSN and convergecast schedule algorithm.

1.2 Assumptions

WSN is assumed to be an ad hoc wireless network of sensor nodes where there is no central server. Nodes are randomly deployed on the tiled plane with one of the nodes being the sink. Nodes are assumed to have location awareness ability (GPS receiver), they turn on their GPS at initialization only (since they are not mobile) and read the location coordinates. They initially have no knowledge of the world and adjacent nodes, they learn of their local world as the algorithm runs.

Definition 3. *Unit Disk Graph (UDG) $I(E, V)$ is a weighted graph where for any two points $p \in E$ and $q \in E$, an edge $e(p, q)$ is in the graph if $\text{distance}(p, q) \leq 1$*

where $\text{distance}(p, q)$ is the weight of edge e [6, 7].

We assume a protocol model where a WSN is a UDG. Each node has a transmission zone where it can send messages if the recipient is within that zone. We refer to this zone as a “neighbor range”. A collision occur when a receiver is in range of two transmitters simultaneously sending. **Collision** in networking can be defined as an interaction in a transmission that may hinder the successful reception of the message or corrupt the message in transit.

We assume that all sensor nodes are equipped with same amount of power, which we refer to as “Energy” and any transceiver activity (transmitting or receiving) consumes the same amount of specified energy. A sensor node that exhausted all its power will be considered as dead and would not exist in the network.

Our network is a decentralized network, meaning that there is no central control. DDS and LCAC algorithms are programmed to run in all nodes on the plane; forming a self organizing wireless network system.

1.3 Summary of Contributions

In our work, we focused on maximizing network lifetime, collision detection and avoidance and minimizing convergecast schedule time. Our work includes the following contributions:

1. A layered and tiled network of sensor nodes (DDS algorithm). We designed a tile network in which each tile has its own tile header (tile sink). Nodes in a tile form a convergecast tree rooted at the tile header. Tile headers form a secondary tree which roots to the Sink. Convergecast tree is formed by selecting dominant nodes in the tile network then a tree is formed from a connected dominating set. Convergecast tree formation is dynamic, meaning on every convergecast round

a new tree is formed based on residual energy and neighbor degree of nodes in a tile. By convergecast round we refer to one complete collection of messages from all nodes to the sink.

2. A local collision free model. This is a collision detection and avoidance model where a node senses if there any nodes transmitting in its transmission zone or if its parent (Destination) is available to receive its packet or it is receiving other node's packets. It is a local model because all nodes sense their transmission zone only, as a result the network operates in a medium with less collision.
3. A one way convergecast schedule algorithm (LCAC). LCAC is a message forwarding schedule algorithm run by nodes which uses DDS tree to collect data to sink.

The performance of our algorithms are evaluated and the results presented show that our algorithms had performed well in minimizing schedule runtime, minimizing collision, load balancing and maximizing network lifetime.

1.4 Organization of Thesis

In the rest of this document we will present the following: Chapter 2 describes types of convergecast, properties and importance of convergecast in WSN. Chapter 3 reviews existing work in convergecast in WSN. Chapter 4 presents the tiled network and DDS algorithm. Chapter 5 gives a detailed description of the LCAC algorithm and the local collision model. Chapter 6 presents simulations evaluation of algorithms and we discuss performance analysis of algorithms. Chapter 7 concludes the thesis and presents future work to be done to extend our work for better performance.

Chapter 2

Background

In this chapter we give a detailed description of convergecast as a type of data collection scheme in WSN. We define and explain types of convergecast and properties of convergecast . And lastly we describe different design objectives, constraints and assumptions used in this convergecast schedules in WSN.

2.1 Convergecast in WSN

Data collection in WSN is an important operation for better analysis and results in any type of WSN. Data collection in WSN can be implemented in many different ways. Initially a sensor would send its packets straight to the sink despite the distance and traffic. Sensors would use enormous amount of energy to transmit over a long distance to the sink.

With the emergence of wireless ad hoc network technology sensors were able to rely on each other to complete data collection by sending packets to their neighbors. Neighbors also forward packets until their messages are delivered to the sink. This algorithm is known as flooding. The relayed communication was a broadcast. Each node broadcasts its packet to the network and the packet is received by all sensors in the range of that transmission. This approach caused high traffic of packets. Sensors

were using their energy to broadcast redundant packets on the network resulting in increasing latency and delay.

Advancement in electronic technology facilitated the use of wireless Ad hoc network in collecting data in an orderly and efficient manner. Sensors now have processing capability and location awareness or neighbor discovery technique to be able to send packets towards the direction of the sink and have minimal routing abilities. The process of sending packets from sensor nodes towards the sink node from all the sensor nodes by using Wireless ad hoc environment is known as *Convergecast*. Convergecast in sensor networks is a many-to-one communication where all nodes send to the sink (root node) [4]. Convergecast transmission is a reverse of broadcast transmission, as seen on figure 2. This can either be through a multi hop manner or by a one hop transactions.

Figure 2: Convergecast vs. Broadcast: (a) Shows Convergecast flow to sink (b) Shows broadcast flow from Sink

2.1.1 Types of Convergecast

Convergecast can be of two types, *raw convergecast* or *aggregated convergecast*. In raw data convergecast, data from every node is sent to the sink as individual packets while in aggregate convergecast a node merges its data with the one it received from

its neighbor and sends it as one packet transmission.

Raw convergecast is the type of convergecast where each node participating in data collection sends its packets either straight to the sink node or uses a multi-hop communication to relay its packet to the sink. All sensors in the line of transmission will forward each individual packet they receive until all its neighbor sensors have completed sending.

Aggregate convergecast uses multi-hop communication to send its packet in a reverse-tree or funnel way where a node close to the root acts as a parent of a set of nodes [3]. The parent node waits until it receives all its children's packets and merges them into one packet and sends it to its parent.

In figure 2, for raw convergecast; sensor node E sends packets of sensor node G after it receives them then sends its own packet on a different time slot. While in aggregate convergecast, node E waits until it receives node G's packet then it appends to its data and sends it as one packet to sink.

2.1.2 Properties of Convergecast

Convergecast in WSN data collection has the following properties [4]:

1. **Convergence transmission**, all nodes forward their messages in the direction of the sink.
2. **Single transmission per node**, In one round a node is to transmit only once and waits for the next round.
3. **One way transmission**, Flow of messages is from sensing nodes (leaf nodes) to the sink.
4. **Collision free transmission**, A packet is transmitted from one node to another in a collision free media.

2.1.3 Why using Convergecast

Over time researchers have come up with different convergecast schedules for WSN. Optimizing different data collection factors like collision minimization, latency reduction, energy efficiency, mobility support, load balancing and fault tolerance can be used in designing a convergecast schedule. A common goal in most schedules proposed is to minimize schedule time. This is because schedule time is proportional to energy efficiency, latency and delay. The following are some of the reasons to use convergecast in a WSN:

- Minimizes Data collection time

Convergecast has lowered the overall data collection operations time to a minimal level. More room to further improve time of convergecast is made possible by advancement of hardware inventions:

- Microprocessors have become more powerful and shrunk in size .
- Memories have evolved to have faster read/write speed and more capacity per size. Process registers can have up 64 bit space, caches can reach up to 3Mb and RAM (Random Accessed Memory) can reach up to 16Gb depending on what system word used (32 bit or 64bit) and hardware integration [8].
- Wireless communication technology has improved to have wider coverage range and even have a dual channel communication.

Research has proved that real time systems like Nuclear leakage monitoring system using convergecast as a data collection method has high reliability due to its fast and reactive response in collecting data as needed.

- Saving energy

Convergecast schedules assign time slots to sensors during which they send or receive packets. Transceivers are powered on only during sending and receiving of packets. Unlike in traditional data collection schemes where sensors have to power up their transceivers all the time to listen and send packets. Research has shown that more than 50% of the sensor's energy is consumed by transceivers activities [1]. Convergecast schedules save nodes' energy and prolong network life time.

- Reduce traffic

The use of aggregate convergecast reduces the number of flooded packets in the network, resulting to reduced traffic in the network. With low traffic in the network, network delay and schedule time are minimized too.

- Improves QoS

Convergecast can be applied to MAC or Routing layer networks to have acknowledgment packets, resend algorithms and fault tolerance mechanism in transmission failures to improve the quality of service in the network.

2.2 Classification Approach and Methodology

2.2.1 Design Objectives

Different convergecast schedule algorithms are designed for different purposes and are being applied to different network systems. In this section we describe a classification of different design methodologies used in most convergecast schedules in WSN. We will give a brief description of the design constraints and assumptions used in the algorithms looked at.

1. Minimizing schedule length

Data collection time in WSN is the main determinant of a schedule's efficiency and performance. A schedule which can run in a short time and manage to collect all data from sensors to sink is said to be an efficient algorithm. For most convergecast algorithms, TDMA is being used to minimize time. TDMA stands for Time Division Multiple Access. TDMA is a sequence of allocating time slots to different nodes on the network so as to have multiple nodes sending and receiving at each time slot [4, 9–13]. Each node is allocated a certain time slot where it could either listen for a packet or send a packet. Once that time slot is over a node turns off its transceiver and sets itself to sleep mode. Nodes save energy while waiting for its next time slot. Concurrent transmission in TDMA can occur when two or more nodes transmitting do not have their transmission zone overlapped or are in different time slots. With a well structured network TDMA maximizes concurrent transmissions and hence minimizes schedule time. In [4] TDMA based schedule is used for collecting data from a tree based network of nodes where it gives priority to more weighted nodes (nodes with higher number of children). Children nodes in all the high weighted parent nodes which are not within their transmission range can transmit concurrently.

TDMA based schedules also aid in energy saving strategies since a node will only fire up its transceiver when its up to either listen or send a packet. In [13] a TDMA schedule is proposed where they proved that their sleep schedule results in up to 50% of energy saving. A combination of TDMA and CDMA (Code Division Multiple Access) can be used in deriving and forming a schedule which uses both contention-free protocols.

Other approaches to design schedules apart from using TDMA are seen in [5, 14–17]. In [5] Rapid Spanning tree (RSPT) is used in modeling a tree based cellular network of wireless sensors in agricultural field monitoring. They used

a Cluster Head Routing (CHR) approach to aggregate data to the sink. In CHR nodes with the highest percentage of energy in their cells are assigned to be cluster heads. Cluster heads receive packets from their children, aggregate the received data and forward the messages to sink. CHR-RSPT proved to be faster than the normal Spanning tree (CHR-SPT).

The use of heuristics in minimizing schedule time is seen in [14]. Authors propose a WSN with a mobile sink. Data collection in this network is performed by the mobile sink by applying racetrack heuristic. Racetrack heuristic is modeled as a Travel Salesman Problem with Neighborhood (TSPN) where in every round, a mobile sink is supposed to cover all nodes and collect data by using the inner line of route (Like race cars negotiate a corner). The Racetrack algorithm has proved to achieve a good approximation ratio.

Neural Network is applied to convergecast algorithms in [15] to implement efficient routes. Hopfield Neural network algorithm (HNN) divides the network into clusters and finds cluster heads then finds sets of paths between cluster heads and sink.

A coordinated convergecast algorithm in [16] uses inward delay and outward transmission delay in the network to form a shorter pattern at which nodes can send packets to their parents. This methodology is used to forward messages all the way to the sink.

Collision Detection can also be modeled into forming faster algorithms like in [17]. Distributed Convergecast (DC) algorithm in [17] uses collision detection capability in a distributed environment to create a flow of nodes to send packets to the sink. DC algorithm has an expected running time of $\log n$.

2. Aggregation means

Data collection in WSN can either be a convergecast or aggregated convergecast.

In aggregated convergecast, sensors forward their packets to their parents or neighbors. A sensor appends all messages it received to its own message and forwards as one packet. This goes on until the packet(s) reach the sink. As seen in figure 2 for aggregated convergecast, node C merges packets received from node A, B and its own and sends it as one packet. This algorithm is also known as input collection. It has been seen that most schedules use Tree-base approach [4, 5, 9, 10, 12, 13, 17] to set up nodes for aggregation. With the sink node being the root of the tree, aggregation starts at leaf level. Packets are forwarded and appended all the way to the root. Tree based aggregation has proven to work well with TDMA in minimizing schedule time.

Other aggregation means seen in convergecast algorithms are the use of solution set [3]. Based on certain parameter of the network (distance, energy) each node selects a neighbor node to forward its packet to. As a result, a set of node pairs is formed which is the aggregation set for convergecast schedule. Breadth First Search (BSF) is also used in aggregating packets for convergecasting.

3. Minimizing latency

Latency in networking is the time delay between when a packet is sent from the originating node to when it is delivered to a destination node. This is usually caused due to queuing in packet buffering or bandwidth management. In WSN, latency is mostly caused by aggregating data over the network or multi hop transmissions. [14] proposed an algorithm for which a mobile sink can pass to all the nodes and collect data in $\log n$ using racetrack heuristics. [11] also proposed a WSN with both mobile and static sinks with the aim of reducing traffic and data loss in the static sinks. Simulations in [11] showed that the more mobile sinks you add in the network the lower latency of data transfer.

Implementing multiple sinks also reduces latency in the network. [18] introduces

P-NLB (Partition-Based Network Load Balanced Routing), a cross-layered protocol for data collection in WSN. P-NLB implements multiple sinks in the network and uses MAC and network layer in routing data. The load balancing capabilities of P-NLB facilitated the system to have minimum latency.

4. Minimizing energy consumptions

Network lifetime of a sensor depends on its power management technique and its transceiver activities. More than 60% of the energy is used by its radio transceiver [1]. Researchers are trying to model and propose networks and protocols that would optimize and increase network lifetime. Efficient data collection algorithms can result into longer network lifetime. [4,5,9,13,14,17,18] have incorporated this energy-saving approach in their algorithms. They got results that show that with efficient convergecast schedules, more energy is saved and increases the network lifetime.

5. Maximize capacity

Capacity in WSN can be referred to as the rate at which data can be sent to sink. One way to implement this is either to increase the rate at which sensors send their information to the sink by increasing their transceiving abilities and have a collision free and high performance network. With the ergonomics of hardware technology, sensors are made to have limited energy to be able to optimize their capacity. But it can be worked in the other way around by increasing processing power, adding mobility to sink or even having multiple sinks on the network. [11,18] have purposed networks with multiple and mobile sinks with the aim of load balancing and increase capacity of data transfer.

6. Maximizing fairness

Fairness or Load balancing in WSN is an important objective to be considered

especially where information collected is of high value. Having a system where all sensors have a fair transceiving fraction of time improves data quality and minimizes packet errors. Parent nodes may be given more transceiving slots because of aggregation of data. WSN with fairness or load balancing schemes [4, 9–11, 15, 18] have proved to increase the throughput of algorithms and minimize latency.

7. Joint objectives

An ideal WSN is one that would have a convergecast schedule that considered all or most of the design objectives and still perform well when compared with algorithms with individual objectives in consideration. Some of the few algorithms proposed integrates these objective. Evaluation is done by comparing them with standard individual schedules. Some of the hybrid schedules seen are Multi-path schedules [11, 18], Energy efficient schedules [5, 9, 12, 14, 16, 18], Mobility support [11, 14] and fault tolerant [11, 18].

2.2.2 Design Constraints and Assumptions

In this subsection, we give a brief description of assumptions and constraints that are generally used in this field of research.

1. Communication and interference model

In Wireless networks we have two network models used in research and designing purposes, these are the protocol model and physical model. Protocol model is the one that makes use of an ideal network. For two nodes to communicate, they should be within their transmission range and only one node sends a packet at a time. If two or more nodes send a packet to a receiver node at the same time and the receiver node is located on both sender's transmission zones then that

is interference and no packets will be received by any node in the zone. In this model the network is usually modeled as a graph $G(V,E)$ where vertices $v \in V$ are nodes and edges $e \in E$ are connections of channels between nodes. Protocol model also assumes to have same distance between nodes so it is normalized to a unit distance between nodes and thus the network is denoted as a Unit Disk Graph (UDG). Protocol model is mostly used for mathematical and theoretical research.

The physical model, as the name defines is the model that takes into account the actual real environment. Factors like interference, permeability of the media and randomization of distance between sensors are considered in physical model. It is also known as the SINR (Signal Interference to Noise Ratio) model. SINR model is used to give realistic effects of how network behaves and how applications of different parameters can affect it. Signals from a sender fades as it propagates throughout the network. The model also takes into account media noise and obstacles that hinders an ideal message transmission. If a signal received by the receiver node is above some reception threshold which considers the environmental noise and obstacles then that is a successful transmission.

2. Types of data collection

There two type of data collection used in WSN, a raw-data convergecast and aggregated convergecast. In a raw-data convergecast packets from nodes are sent individually to the sink either through single or multi hop transactions while in aggregated convergecast packets are appended in higher nodes packets and being sent as one packet. The appending of packets follows the hierarchy in the network. For energy efficient and analytical model, aggregated convergecast is suitable to use but for a real time and sensitive model a raw-data convergecast is suitable.

3. Network topology

Wireless networks have different topology based on where they are applied. It can either be a line, star, mesh or tree topology. Most schedules used tree topology as it seemed to work well with aggregating and minimizing schedule time. Some algorithms have a dynamic topology where sensors are mobile. Nodes choose destination nodes of their packets instantly based on some factors of the network (distance, energy or speed) [11,18].

Network Layers are also taken into account in WSN. Most algorithms are based on **MAC layer** or **Network Routing Layer** of the TCP or OSI model. In MAC layer, sensors use broadcast transmission to transfer data. All neighbors of a transmitting sensor receive the message and they also transmit it in a broadcast manner. In Network routing layer, address-specific forwarding of message is used. A sensor sends a packet towards the sink using a specific routing channel and only the designated recipient responds to the transaction.

4. Sensor deployment

Sensor deployment in WSN is how sensors are set into the field. Most algorithms assume random deployment or grid deployment where in the latter, a sensor is placed in the grid of the test field. Some algorithms have mobile sensors. Sensors start off at a common point and they form a network based on the area covered. Others use air-drop approach where sensors are dropped on the on the field by an aircraft [12].

5. Buffer size

Buffer can be explained as temporary storage of packets pending transmission or pending analysis. In WSN some networks assume unlimited buffer size in sensors and sink while others set a limit to sensors to simulate the realistic

ergonomics of hardware.

6. Transceiver assumption

In most Wireless sensor networks, a sensor is equipped with one omni-directional transceiver antenna which has the capability of transmitting and receiving packets on a specified channel. Sensors are designed to control when to turn on the radio amplifiers so as to save energy when it is not receiving or sending.

7. Implementation pattern

In WSN, data collection algorithms can be implemented as a centralized or autonomous process. In centralized WSN, either the sink or a dedicated node is set up to allocate time slot codes and aggregate nodes into a tree structure or cluster. While in autonomous WSN, each sensor is programmed to learn the environment and advertise itself to its neighbors. Eventually nodes learn who are its neighbor nodes and where the sink is located. Then from the collected information, a convergecast pattern is formed.

Chapter 3

Literature Review

Over the years, there has been a lot of research on the field of wireless sensor networks which resulted to different convergecast schedules. These schedules differ in their objectives, constraints, assumptions and type of system it was designed for. We looked on several algorithm for which most of them had a common goal of minimizing schedule length using Time Division Multiple Access (TDMA) approach.

In this chapter we give a description of some of the existing convergecast schedule algorithms to give a clear understanding of what has been done so far. We will present algorithms based on their main objectives starting with minimizing schedule length which is the most common objective used to those with joint objectives. When describing these papers, we classified them according to the different design objectives.

3.1 Minimizing Schedule Length

WSN convergecast algorithms are of two types, raw data convergecast and aggregated convergecast. A description of raw convergecast algorithms will start then followed by aggregated convergecast algorithms.

Raw-data convergecast: In raw data convergecast schedules all packets from sensors are routed to the sink without any aggregation. The following algorithms

implement raw-data convergecast:

In [12] Kam and Schurgers introduced a hybrid MAC/Routing convergecast protocol for a network of a UAV (Unmanned Aerial Vehicles) which uses long range radio frequency for communication. In figure one of [12] a graph of connectivity vs. transmission is presented. With transmission range less than 80 meters some nodes do not have a route to the sink. So they came to a conclusion that in long range radios it is best to have at least a 80 meter radio range between nodes. Most of the nodes are one hop away from the sink. For the few which are not in one hop reach to the sink, a routing functionality was incorporated to the MAC layer so as they can be able to send their information to the sink. Thus, a hybrid MAC/routing convergecast protocol called ConverSS.

ConverSS is a two phase approach which starts with a one hop MAC transactions then the second phase is for the multi hop nodes. In phase one, all one hop neighbor nodes send their packets to the sink. The sink replies with an ACK packet. The algorithm uses Time Division Multiple Access procedure (TDMA). Nodes which didn't receive the ACK packet from the sink are then moved to phase two. In phase two, all the packets that didn't receive an ACK from the sink flood their packets in the network. Eventually the packet will reach the sink. The sink broadcasts a Delivery Status Message (DSM) which with multi hop transaction it will reach the respective node. Phase two is the multi hop network phase of ConverSS.

To implement these two phases, ConverSS has some features that manage packet flow, latency and energy efficiency strategies. ConverSS has a header in all its packets. The header is used in routing the packets, counting number of hops from sink to the sensor nodes. It is also used to state the type of message if it is a payload, ACK or DSM. ConverSS uses immediate delivery confirmation (ACK) and implicit routing for flooding nodes to control flooding of packets in the network. In the latter option, if a node has a parent or it is a child it forwards its packets to the parent instead of

flooding the packet on the network. To have energy efficient listening and sleeping schedule, ConverSS uses Delivery Status Bitmap (DSB) to schedule what time to wake up and listen for packets.

Grandham et al presented their *Minimal time convergecast schedule* in a distributed environment in [13]. They first used a linear network in deriving the algorithm then they applied it to a multi-line network. They proved that tree networks can be reduced to multi-line networks.

In [13] it is argued that a guaranteed packet delivery scheme like CMSA (contention free) is reliable but has high latency and hence has delayed schedule while a TDMA contention based scheme can be used to eliminate collision and yet bound on the time required to complete convergecast. Distributed minimal time convergecast algorithm is a distributed convergecast algorithm that uses TDMA and requires at most $3N$ time slots to complete a convergecast. This algorithm has a sleep schedule that saves up to 50% of the network energy.

Grandham et al used two types of network topology: linear network and multi-line network. A linear network is the one which the sensing nodes and the sink are all connected in a linear topology. A multi-line network is the type of network that has subnetworks in clusters which are connected in a linear network with the sink. Each cluster has a linear network of sensors which has a cluster head that connects to the sink.

For linear networks, a node has to be in one of the three states: T (transmitting), R (Receiving) or I (Inactive). Nodes switch states in every time slot. All nodes obtain their hop count from the base sink before the schedule starts. A node with hop count h is assigned a state according to the *mod* function (Modulus). If $h \bmod 3$ is 0 then it is assigned state T, if $h \bmod 3$ is 1 then I and if $h \bmod 3$ is 2 then R. Figure 1 and 2 in [13] shows the convergecast schedule and state transition diagram respectively. This approach has been proved to use a lower bound of $3N - 3$ to complete convergecast

where N is the number of nodes in the network. Mathematical induction was used to prove that it can further be lowered to $3N - 2$ in Theorem 2.2 of [13].

For multi-line network, the same approach was used to all the linear networks of the multi-line network forming parallel transmissions in the network. The schedule was then designed to accommodate multiple transmissions to the sink. In each time slot at most one line branch should send its packet to the sink. At the initialization stage, the sink assigns a unique ID to every branch. Each node gathers important information about its branch: branch ID, hop count and number of nodes in its branch. The network has two layers of scheduling taking place simultaneously; branch heads use their ID and slot to send packets to the sink. At the same time nodes in the respective branches transmit their packets toward their branch head thus in overall less time is used for convergecast. The lower bound for the number of time slot in the multi-line network is proved to be $\max(3n_k, N)$ where n_k is the maximum number of nodes in a branch.

Zhang et al in [16] extended the convergecast framework by incorporating an inward delay mechanism. Before it considered only an outward delay mechanism. In an inward delay mechanism, nodes farther away from the sink start to send their packets. Inner nodes wait for the packets to reach them then they forward theirs too to the sink. In an outward delay, packets closer to sink start transmitting while the ones farther wait till all the packets inside transmit then they transmit too. This was based on the temporal coordination heuristic which instead of a node to just send a packet immediately, a node v has to wait for time T_v before sending. This was expected to reduce congestion around the sink node during convergecast [16]. With a hop count distance h_v of a node v to sink, a time function $T(h_v)$ is used to calculate the respective waiting time of each node.

Equations 1-5 in [16] shows how the $T(h_v)$ are derived for both inward and outward delay mechanism. However, with these functions; nodes which are farther away from

the sink will end up sending packets at the same time and therefore at some point in the process congestion will form at the sink. Zhang et al proposed an addition of a random offsets which are uniform random number $[0,1]$ variable r and dh which is number of nodes at distance h where d is network density. The resulting equation is of the form $T(h) = t(h) + rdh\tau$ where $t(h)$ is the original time function (equation 2 and 4) and τ is the time for one hop transmission.

A layered routing protocol is introduced in [16] implemented convergecast in a layered approach. The routing protocol had the following layers: application, initialization, convergecast, delay-transmit, transmit-queue and MAC layers. The application layer generated process instances and scenario of the network. Initialization layer setups and initializes nodes sink by flooding packets to broadcast hop counts to the sink. In convergecast, there are two ways: by using convergecast tree or directional flooding to sink can be implemented. For [16] directional flooding was implemented. In Delay-transit: This layer controlled time of transmission of packets. It uses a monotonic non increase probability function p to regulate retransmissions in the network. If the packet is flooded in the network it does not get retransmitted again [16]. Transmit-queue layer maintains a buffer of packets to be sent or received when the radio transceiver is busy.

Layers that can be implemented to improve the framework to be used for aggregation convergecast protocol are *duplicate-transmit* and *aggregate-queue* [16].

In [9] CTCCAA (Convergecasting Tree Construction and Channel Allocation Algorithm) was introduced. CTCCAA creates a convergecast tree which can also be used for broadcasting. CTCCAA uses greedy approach in tree construction and channel allocation. It creates groups (parent and child) on trees based on BFS (Breadth first search) and distance from root and parent. CTCCAA was a solution to CTC-CAP (Convergecasting Tree Construction and Channel Allocation Problem) which was how to construct a tree and assign each node i a code called PCC(i) and a slot

called PCS(i) to communicate with its parent node, thus the parent uses the PCC of its children to receive packets.

The following are constraints which were considered in [9]:

1. If the children of one group are within the transmission range of another group's parent then the two group's children must have different PCC(i) for communicating with their parent or they should have different set of slots (PCS(i)). This is because if they share the same PCC then nodes of group one who are closer to the parent of group two might be in confusion in sending to group two's parent, also having different PCS (i) will enable the two groups to operate in different time slots, so there will be no confusion. This was referred to as co-channel criterion and demonstrated in figure 3 of [9].
2. If $dist(node1, B) < dist(node2, B)$ then $PCS(node1) > node(node2)$ else $PCS(node2) > PCS(node1)$ [9]. This ensured that nodes which are closer to sink are parents and are given a slot that is less than their children's so as it receives packets from its children first then it sends after.
3. No two children of the same group share the same slot to communicate with the parent but they share the same code (PCC).
4. The last constraint considered in [9] was children of each parent are chosen based on shortest distance to its parent (proximity criterion).

In CTCCAA, channel allocation starts from the root node and proceeds all the way to the leaf nodes. A node using CTCCAA broadcasts a packet with channel allocation to all nodes. The algorithm maintains two lists, currentlist and nextlist. When it runs for the first time, the currentlist are all the nodes in level 1 which is the root node and the nextlist will be empty. It uses proximity criterion and co-channel criterion in allocating slots and codes to nodes and uses the following definition in

identifying interference nodes (IL_k).

$$IL_k = \{s \in V : neighbor(s) \cap neighbor(k) \neq null\}$$

Figures 4, 5, 7 and 8 in [9] simulates the tree construction, code and slot allocation of CTCCAA. Nodes are aligned to parents based on the proximity criterion. Allocation of codes and slots is performed by considering the constraints of the system. Figure 6 gave the convergecast tree construction and channel allocation algorithm. Simulations where done on randomly generated graphs by varying the number of transmission code and node density. Running times where compared with those ran on broadcasting trees in figure 9 of [9].

Racetrack approximation algorithm in [14] tried to solve a mobile sink routing problem in convergecast by implementing TSPN (Travel Salesman Problem with Neighbors). It proved to complete a TSP route within $O(n)$. Racetrack algorithm is derived from a protocol model network where all sensors where Unit disks. A mobile sink had to find the shortest path to pass to all the sensors once (TSP) and collect data. Figure 1 in [14] simulates the Mobile Sink (MS) path in the network with and without the radio range of the sensors. The two heuristics; Auto racing track and shortcut finding heuristics made the MS convert its route by considering radio range of the sensors as shown in figure 3.

The goal of racetrack algorithm was to find the shortest route T_{MS} which starts and ends at the point S and travels through all disks either on the edge or within the disk. i.e. $\forall i(1 \leq i \leq n), dist(V_i, T_{MS}) \leq r$ where V_i is the center point of disk i . This was implemented by finding TSP route (T_v) in the graph network then applying racetrack approximation and shortcut finding heuristic to improve it to TSPN (T_{MS}). Figure 2 of [14] shows a MS route passing through the center of the disks while figure 3 displays the inner route of MS which is introduced by heuristic 1. Heuristic 1

Figure 3: Mobile sink using TSP to navigate to all sensor nodes in a shorter route by getting in the radio range as shown as a UDG

states that in a racing track, the inner lane is the shortest path than the outer lanes. Substitute points, ISP (Inner Substitute Points) where found on the boundary of the disk and a centroid C was calculated out of ISPs. A line V_iC where V are vertices or center of disks to the Centroid determined the ISP of each disk, denoted as I_i .

$$Centroid(C) = \frac{1}{n+1} \left(\sum_{i=1}^n (V_i + S) \right)$$

Theorem 1 in [14] states the rules to choose the ISPs which are for any ISP I_i :

1. if $dist(V_i, C) \geq r, I_i = R_i$;
2. if $dist(V_i, C) < r, I_i = C$,

where r is the radius of the sensor disk and R_i is the point where the segment V_iC intersects the boundary disk. Proof of theory one was done in a polar coordinate system. Figure 6 gave shortcut finding algorithm, a dynamic programming algorithm which is using the zones with overlapped radio ranges to re-route the MS. And at the end proof of approximation ratio of Racetrack algorithm was given.

Aggregate Convergecast: In aggregate convergecast schedules, sensor nodes merge their data with those received from its neighbors or children then sends it all as one packet.

An approximation algorithm for data aggregation was introduced in [2] by Huang et al based on *maximal independent sets* with a latency bound of $23R + \Delta - 18$ where Δ is the maximum degree of sensor nodes and R is the network radius. The network model in [2] was a UDG $G = (V, E)$ where $v \in V$ are sensor nodes and an edge $e \in E$ denotes a connection between nodes. $e \in E$ exists if and only if the distance between them is at most 1. Data aggregation in [2] is viewed as a series of senders $\{S_1, S_2, \dots\}$ where at each time slot k all the data collected from $\{S_1, S_2, \dots, S_{k-1}\}$ are aggregated to nodes in S_k where $1 \leq k \leq l$. l is also referred to the *data aggregation latency*.

Data aggregation tree construction in [2] used maximal independent set of nodes to divide the network into layers (from sink having layer 0) and each layer had an independent set of nodes of another layer. This was implemented by applying a coloring scheme. Sink starts with BLACK color and then all nodes on layer 1 are colored WHITE (non-black). This goes on all the way to the highest layer in the network. For those nodes, say at layer l_i whose layers l_{i-1} and l_{i+1} are both black then they are set to WHITE. Each WHITE node in layer 2 with BLACK children in layer 3 are marked BLUE and the parents of these BLUE nodes in layer 2 must be BLACK. An edge is drawn between the layers of independent set (BLACK, BLUE and WHITE (non-black and non-blue)). Algorithm 1 and 2 listed in [2] implement the aggregation tree in initialization and construction.

Data aggregation schedule in [2] uses a tree constructed using maximal independent set. A set X is created with nodes which are picked from different senders set $\{S_1, S_2, \dots\}$ and each node is checked if it conflicts with all the other nodes in that time slot. All nodes that are in set X and have no conflict are added to the permanent transmission set W of that time slot. The process goes on for all time slots and all the

nodes are scheduled. Algorithm 3.1 in [2] presents the First-fit scheduling algorithm which does all this. Algorithm 4 applies the process of algorithm 3.1 by setting the different sets in the network (WHITE, BLUE and BLACK) as its input and repeats it in a pattern that data collection is performed from leaves to the root.

Gupta et al in [10] also used TDMA/CDMA in scheduling data collection in Tree-based network of sensors which is formed based on distance of nodes from the source (sink). [10] modeled its algorithm in a weighted graph network, where the weights signify cost of communication between two nodes which refer to distance. A low latency energy efficient algorithm for convergecast was introduced in this work. The main parameters of the algorithm are latency, energy saving and reliability. The problem addressed in [10] is how to come up with a system that has minimum convergecast schedule length and yet minimize energy consumption of the system. Energy used by a node i is a function of amount of data transmitted d_i and distance of transmission r_i denoted as $E_i = f(d_i r_i)$. In a single hop communication environment the distance of transmission is big but overall energy usage is low while in a multi hop communication the distance of transmission is small but overall energy use is high because of having many transmissions. Gupta et al justified that either mode of communication is important in WSN since they all tend to balance out in the $E_i = f(d_i r_i)$ equation. Equation 1 in [10] was used in this explanation. [10] used multi hop communication and not single hop because in free space, energy distance (transmission range) relation is quadratic while energy to data bits transferred is linear. So with multi hop, the transmission range is lower and so saving energy in the system.

The tree construction algorithm in [10] (algorithm 1) starts at the root node which is the sink and works its way up to the leaves as it classifies nodes into parent-child relationship. The factor used in this classification is the weight of the graph and the β -rule. β -rule states that *A node cannot have more than β children*. The β rule

made the tree more balanced and hence maintains average energy level overall the network by avoiding overloading few nodes. Figure 1 in [10] demonstrates the tree construction algorithm.

Gutpa et al in [10] applied channel allocation to the network by taking the constructed tree and communication codes then proposed an algorithm to allocate channel codes to nodes. Any parent node in the network has the reception code same as its children's transmission code. Algorithm 2: NodeInfo in [10] is a procedure that checks distance between two nodes u, v and their parents ($parent(u)$ and $parent(v)$). If node v is in the neighbor of u ($Adj(u)$) and $\delta(u, v) \leq \delta(u, parent(u))$ (which are distance between u and v and distance between u and $parent(u)$ respectively) then node v is added to a receiving set Π_i . Which means that one of their neighbor uses code i to transmit, which for this instance is the children of u are transmitting to u since v is closer to it. v can hear u 's transmissions. Therefore v and u can both hear children of u , and if v is in a neighbor of $parent(u)$ and $\delta(v, parent(u)) \leq \delta(v, parent(v))$ then v is added to a transmission set π_i which means that one of the member is transmitting with code i ; which is node u transmitting to $parent(u)$.

Algorithm 3 in [10] which is Channel allocation algorithm, takes in a set of nodes which are not assigned channel codes and re-runs algorithm 2 with the set ℓ_{new} which are unassigned nodes. Also it checks the parent-child pair if for a code i ; $u \notin \pi_i$ and $parent(u) \notin \Pi_i$ then the two can use code i to communicate.

Kumar et al in [15] looked at the convergecast and energy saving problem in a different approach compared to the common ones. They proposed the Hopfield Neural Network (HNN) algorithm in which they modeled a WSN as a Neural Network (NN) with the following steps: Division of sensors networks into clusters, set a path from each cluster heads to sink by using reliable sensor nodes in the network and create a convergecast tree from these paths using HNN.

The network model in [15] is a undirected graph $\xi = (S, E)$ where S is a set

of sensor nodes and E is a set of edges representing links between nodes. The set $Z \subseteq S$ represents nodes which participate in convergecast, each node in Z has at least an average energy level to be able to participate in convergecast. Kumar et al used the idea of common links and energy levels between different nodes in choosing convergecast routes. The objective was to choose paths with minimal common links and overall maximum energy level. Figure 1 in [15] demonstrated a network of cluster heads with n_1 as a sink node. Each node n_i have energy value p_i .

Tree creation in [15] takes place in three steps; Step 1: Finding cluster head in the network using Kohen self-organizing Neural Network (KNN) model. Step 2: Finding a set of sink-node pair and Step 3: Construction of a convergecast tree.

In step 1: KNN [19] takes rows of adjacency matrix of the network and it trains on them and forms clusters due to similarities in patterns. Procedure 1 in [15] describes this implementation.

In step 2: This involves computation of paths from sink node to sensor nodes and considers the path's overall energy level and minimal common links.

In step 3: Convergecast tree pairs are selected from the sink-node pairs created in step two. The following notations where used: $(s_1, s_d)_i$ is the i^{th} node pair with s_d as a destination node and $d \in [2, N]$, $NP(i)$ is the number of paths in the i^{th} sink-node pair [15]. V_{ij} and Y_{ij} are the j^{th} path of i^{th} sink-node pair and binary variable to determine if V_{ij} is taken or not (1 if taken and 0 if not taken). To implement an effective and energy efficient convergecast tree, Kumar et al proposed the minimization of the following objective equation:

$$\frac{1}{2} \sum_{i=1}^{N-1} \sum_{k=1}^{N-1} \sum_{j=1}^{NP(i)} \sum_{l=1}^{NP(k)} f(V_{ij}, V_{kl}) Y_{ij} Y_{kl}$$

for $(i, j) \neq (k, l)$

where $f(V_{ij}, V_{kl}) = [(V_{ij} + V_{kl}) - (E_{ij} + E_{kl})]$, V_{ij} is the number of links in V_{ij} path.

$(V_{ij} + V_{kl})$ is the number of common links shared by the two paths.

$(E_{ij} + E_{kl})$ is the energy levels in the two paths. Procedure III in [15] describes this process. Kumar et al implements a convergecast tree by applying these three steps in a neural network model (KNN) which uses Neurons as inputs to the network. Each neuron is specified with a certain parameter and is connected to all output neurons [15]. For HNN, one neuron was defined for each path between every node pair. So the number of neurons created for the construction of convergecast tree is $\beta = \sum_{i=1}^{N-1} NP(i)$ [15]. A learning algorithm was run in this input neuro-network and based on pattern similarities it forms the output network tree.

Using distributed environment in [17], Kesselman et al introduced a Distributed aggregate Convergecast (DC) algorithm which uses transmission power of nodes as a parameter. Their algorithm runs in $O(\log n)$. Based on a Signal Power-distance relation $Power(P) \propto (Distance(d))^{-\alpha}$ the Distributed Convergecast (DC) algorithm determines transmission range $R_u^t = (P_u^t)^{\frac{1}{2}}$ of each node u in every time t . P_u is the transmission power of u . In a network denoted by a directed graph $G^t = (V^t, E^t)$ which is a graph captured at every time t . V^t are the active nodes in the network which have energy level which is above a threshold energy required to participate in convergecast. E^t is a set of edges where for node u to transmit to node v , edge (u, v) must be at most R_u^t . DC starts by initializing all nodes and setting the transmission of each node to the closest active node: $\min_{v \in V^t} d(u, v)$. Nodes transmit with a constant probability p . If its transmission is a success then its set to inactive mode. The goal of DC is to minimize latency and still keep energy consumption low. Figure 1 in [17] gave the Distributed Convergecast algorithm. Theorem 3.4 and 3.5 where used to state that DC algorithm runs in $O(\log n)$ and have a lower bound of $\Omega(\log n)$.

Karthickraja et al provided an efficient novel hybrid routing protocol for data aggregation using rapid spanning tree (RST). [5] proposed this self-organizing protocol

based of the agricultural irrigation systems and water resource management issues in Coimbatore, Tamil Nadu state in India. WSN model used in [5] was a clustered network of sensor nodes. A network was made up of M clusters which have n sensor nodes with one of it being a cluster head. A cluster head is selected among the nodes. The selection criteria was; nodes with high energy to be able to transmit in longer distance than other nodes (Trans-cluster transmissions). The protocol has five parts in its operation, which are *Attaching to cluster head*, *Constructing spanning tree*, *Sensing*, *Data forwarding* and *sink*.

In *attaching cluster head*: Initially nodes are formed into clusters by a distributed clustering algorithm. All nodes sends their energy level values to the cluster head. The one with the highest energy level is selected to be the cluster head. A broadcast message is sent within every cluster to apply this configuration change. This procedure is also done when the current cluster head has a below threshold energy level. In *Constructing rapid spanning tree*: A rapid spanning tree topology is applied to the network where there is no cycle in the network. There is only one path from say node i to node j in the network. The proposed hybrid routing protocol is then run on the tree network constructed with key operations being cluster management and cluster head assignment due to topology change or energy drainage in nodes. Also routing and re-routing data in case of path failures.

Chen et al solved the Minimum Data Aggregation Time (MDAT) problem in [3]. They proved that the problem is NP-hard. They introduced a Shortest Data Aggregation algorithm (SDA) which forms a set $\{(S_1, R_1) \dots (S_s, R_s)\}$ where S_s and R_s are sending node and receiving node respectively. SDA runs in $(\Delta - 1)$ where $\Delta + 1$ is the maximum number of sensors within the transmission range of any sensor. Author's in [3] used UDG to model the network. Their algorithm uses a technique of finding a data aggregation tree after the schedule is made.

In a network $G = (V, E)$, SDA takes MDAT instance (G, S, d) as input where S is

a set of nodes possessing information requested and d is the sink node. A Shortest Path Tree (SPT) is generated from MDAT instance (G, S, d) and is used for traversing from its leaves to its root d . Sets Z_r, S_r, R_r and I_r are respectively leaves of T_r , set of senders, set of receivers and set of informed nodes that take part at time slot r . At each time slot a node from $Z_r - S_r$ with maximum number of neighbors is checked if it can be removed without destroying Z_r . Then its added to S_r , else a node from neighbor set which are not in leaf nodes set and not sender set is selected and added to a R_r set. Lemma 3 and 4 in [3] listed some constraints which guide and define some parameters used in the algorithm. Lemma 3 and 4 are also used in theorem 2 to prove that SDA runs in $(\Delta - 1)$.

Nikolaidis et al introduced an aggregate convergecast algorithm in [4] that uses instances of bipartite graph semi-match in tree construction and a weight based scheduling scheme. They improved the tree construction and scheduling by maintaining the original structure of the constructed tree throughout the schedule. Their main objective was to model a tree network which will have a lower schedule length $T_{min} = \max\{|\xi_i| + d_i : i = 1, 2, \dots, N\}$ where ξ_i is the children set and d_i is the distance from root node of node i . Balanced Shortest Path Tree (BSPT) is the tree construction algorithm used in [4]. BSPT takes the layered view of nodes in the network and treats nodes in layer say $d + 1$ children of nodes in layer d whereby layer 1 is the root node. A balanced parent-to-child assignment of nodes is done by finding a bipartite graph match of the two layers d and $d + 1$ such that $\max\{|\xi_k| : \forall k \in C_d\}$ is minimum where C_d represent the set of nodes at distance or layer d . This same process is done for all level in a Breadth First Search until the whole tree is formed as G_b in Algorithm 1 [4] and E' is a load-minimized tree of G_b . First fit algorithm in [4] demonstrates Bipartite graph matching done in this tree construction algorithm described as from [4] on algorithm 1.

A ranking based scheduling algorithm, Weighted Incremental Ranking for convergEcast with aggregation Scheduling (WIRES) was proposed by Nikolaidis et al in [4]. WIRES uses weight $w(i)$ of node i to assign it with a time slot to send or receive data. In their case they used *number of non-leaf neighbor* $\eta(i)$ as their weight metric where $\eta(i) \subseteq N_i$. N_i is the neighbor set of node i . WIRES retains structure of the tree unlike other tree construction algorithms which forms another tree as they schedule nodes. This algorithm obeys and maintains constraints C1, C2, C3 and C4 of the system model in [4]. WIRES takes in a tree network graph as parent-child relationship of nodes and gives out sets S_j and R_j which are senders and receivers set of slot j respectively.

WIRES assigns time slot in decreasing order of weight $w(i)$. Before it assigns a node e' a time slot it first checks if the parent of e' , P_e is a neighbor of any sending node in that slot j . And it also checks if the node e' is a neighbor of any receiver node in the slot j . If a node e' meets all these checks, it is being added to the sender's node set (S_j) of slot j and its parent P_e is added to receiver's node set (R_j) of slot j .

3.2 Minimizing Latency

Latency in networking is one of the biggest challenges since in practical view, it is very hard to minimize it than in theoretical manner. Latency is the measure of delay time between when a packet is sent to when the packet is received. Minimizing Latency plays a big deal in minimizing the overall schedule time, which as seen is the main objective of most algorithms in the WSN data collection.

Erman et al used a multi-sink network and a load balancing routing protocol to minimize latency in [18]. Having multiple sinks in the network reduces the path length between nodes. Having a load balancing protocol (P-NLP) ensures nodes and sinks are having a reasonable number of children hence minimizing latency and overall

schedule length.

In [10], Gupta et al proposed a heuristic solution for minimizing energy consumption in convergecast and minimizing data latency in the network. In the latency model, Gupta et al described latency in convergecast depends on the number of parallel transmission in the network. The more parallel transmission there are the less latency in the network. A balanced tree was modeled with the likelihood of maximizing multiple transmissions in the network and hence minimizes latency. In this model, their goal was to minimize kT_D which is the time for k children to send data to its parents. T_D is the determined length of one time slot which is a maximum time to send a data unit expressed by $D = \max\{d_{c_1}, d_{c_2}, \dots, d_{c_n}\}$. d_{c_i} is a data unit from child c_i .

3.3 Minimizing Energy Consumption

WSN is the type of network which is designed to operate in remote, sensitive or dangerous areas where human accessibility is hard. This resulted to sensors having limited capabilities due to its ergonomics and miniature designs. Power is among the most limited feature since battery cells take a lot of space and weight. In [14] energy consumption is minimized by setting the sink to be mobile to go to all nodes to collect data instead of nodes to send to the sink in a longer distance. Since the sink node is suppose to be more powerful and equipped with faster and more energy storage capabilities, nodes then have minimal energy consumption because they transmit in a very short distance. Nodes don't have to re-transmit its neighbor's packets. The RaceTrack approximation algorithm which uses TSP routes was able to perform this in $O(n)$ computational time.

Erman et al in [18] and Lee et al in [11] applied the logic of *minimizing transmission node's distance* by having multiple sinks in the network as an energy minimization

strategy.

In [5] the Cluster Head Routing- Rapid Spanning Tree routing Protocol (CHR-RSTP), a cluster based algorithm for data collection in WSN was introduced. Cluster head are picked according to the amount of energy remaining in nodes within every convergecast round. This is because Cluster heads had to transmit to longer distance and also listen to transmission of its children. Equations 1 to 4 in [5] shows that energy consumed at cluster head is much higher than sensing nodes.

Gandham et al, in solving the Energy minimization problem they proposed a sleep schedule in [13]. The sleep schedule could save up to 50% of the energy. As seen in the previous section, they used line networks to derive their algorithm. Same concept was applied to tree network which are reduced to multi-line network. The sleep schedule proposed switches off the transceivers of nodes when they are inactive or when nodes have forwarded all the packets. They assumed that the energy spent in Transmit, Receive and Inactive states are the same (e joules). In line network their algorithm uses at most $3N$ time slots to finish the convergecast. With no sleep schedule it uses $3N^2e$ joules. While with sleep schedule applied, the total energy consumption is lowered to $\frac{3N(N+1)e}{2}$.

In multi-line networks, a network of N nodes with k branches with equal number of nodes requires about N time slots to finish a convergecast. The algorithm uses at most N^2e joules without sleep schedule. When sleep schedule is applied as in line network, the overall energy consumption is reduced to $k * \frac{3\frac{N}{k}(\frac{N}{k}+1)e}{2}$ joules.

Distributed Convergecast (DC) algorithm in [17] by Kesselman et al used minimum energy convergecast approach as a decision criteria. Minimum Energy Convergecast algorithm (Denoted as OPT) is the routing of data packet based on minimum energy consumption per transmission i.e. a node u will transmit to the closed neighbor node v since $Energy(E) \propto \frac{1}{d^\alpha}$ where d is the distance of transmission and α is an environmental variable. The minimum energy convergecast algorithm's energy

consumption is denoted by C_{OPT} which is $\min\{\sum_{t=0}^{t_f} E_A^t\}$ where E_A^t is the energy consumption of algorithm A at time t and t_f is the termination time [17]. Claim 4.1 and Lemma 4.2 in [17] proved that DC algorithm is C_{OPT} .

3.4 Mobility and Multi-path Support

In the efforts of minimizing schedule length and latency in WSN convergecast schedules, integration of mobility and multi-path support in networks proved to be helpful. Mobility refers to either mobile sinks or mobile sensors though the latter might be more expensive to implement. Multi-path refers to having more than one sink in the network where nodes can route their packet to the preferably nearest sink.

In [18] Erman et al implemented a multi-path support network and proposed a data collection protocol (P-NLP). This protocol has two routing modes: S-SPM (Smart-Shortest Path) mode and LBM (Load Balancing Mode). S-SPM uses local node information for data collection. It clusters nodes into parent-child relation. LBM uses local node information and global information to route messages. Global information is the node-sink information exchange. LBM is responsible for assigning nodes to sinks which are closer to them.

Lee et al in [11] introduced a WSN communication model and protocol based on multiple sinks and mobile users so as to reduce traffic and data loss at the sink. Nodes pick a closer sink for better energy efficient transmissions by listening for advertisement from sinks. They reply to the chosen sink to bind itself to it. Selection of sinks is based on hop counts which are read from the advertisement packet from the sink.

User mobility management in [11] is implemented by a mobile user having Primary Agent node (PA), Secondary Relay node (SR) and Foreign Relay node (FR). A PA is the node that a mobile user talks to to get all the information of the network. A

FR is a node that is used to relay packets to the mobile user when it is out of its PA zone. And a SR is more like FR but it is a second relay node after FR. This concept is used for both local mobility management i.e. within one sink's domain or global mobility management i.e. when a mobile user switches between one sink domain to another.

In this chapter we covered different schedule algorithms for convergecast data collection on wireless sensor network. Classification of this background review was in layered approach. Algorithms with different design objectives were described. We present our network model and dynamic dominating set construction algorithm in the next chapter.

Chapter 4

Dynamic Dominating Set Algorithm

In this chapter we define our network model where we will run our algorithms and its properties. We present Dynamic Dominating Set algorithm (DDS), our tree construction algorithm. DDS algorithm uses node's residual energy and neighbor degree as metrics for choosing a dominating set.

4.1 Network Model

In our design of a network, we assumed a UDG network of wireless sensor nodes with a sink node to which all sensed information will be routed to it. The main objective of our work is to jointly minimize energy consumptions, minimizing schedule time and reduce collision on the network.

We designed a tiled network on a plane. Assuming random deployment of nodes on the plane, nodes which fall on the same tile will learn of their neighbors and form a connected network of nodes within the tile. We refer to these networks of nodes in different tiles as "Tile Networks". The aim of dividing a network into tiles is to facilitate load balancing of packet traffic on the plane since each tile will have a tile sink. A tile sink will forward the packets to the network sink. Figure 4 illustrates our network model, a two dimensional plane divided into equal tiles. Sensor nodes are

randomly distributed over the plane.

Mathematically, our network model can be defined as follows:

Let a network of sensors in a tiled plane be a UDG $G(V, E)$ and a secondary network of dominating nodes in each tile be $G_D(V_{GD}, E_{GD})$. Let there be a set of M tiles in the plane. This set of tiles is denoted as $T \{T_1, T_2, \dots, T_M\}$ and each tile will be referred to as T_i .

N sensor nodes are randomly distributed over M tiles, say by air drop where $N > M$. Each tile T_i has k_i number of sensors where k_i is the number of sensor nodes in a tile T_i . And so $N = \sum_{i=1}^M k_i$, for our case we assumed all the tiles have same number of sensors $(k_1, k_2, \dots, k_M)$. And so $N = Mk$.

Tile Grid

Figure 4: Tile Network plane with nodes forming a tree in every tile. Tile sinks form a secondary network to connect to the Sink.

4.2 Dynamic Dominating Set

4.2.1 Dominating Set

Definition 4. *Dominating set* D of a graph $G(V, E)$ is a subset of V where every other node in the graph and not in D are connected to at least one member of D [20].

Dominating set in networking can be related to relay nodes or routers, where they are placed at points where they can be reached by most nodes on the network. **Connected dominating set** is a dominating set D which is connected to form a subgraph. Our convergecast tree will be formed by connecting dominating sets of every tile.

4.2.2 Dynamic Dominating Set

Consider tile T_i , let the network graph $G(V, E)$ have a set D_S of vertices from V which forms a connected dominating set. We compute D_S using the following parameters which will be specified on initializing the network;

- Δ_v : Node degree (number of neighbor connected to a sensor node v)
- E_v : Residual energy of a sensor node. Residual energy is the amount of energy a node v has which decreases as it sends messages.

We introduce the **dynamic nature** to our network by having nodes run tree construction algorithm on every time cycle. Unlike BSPT in [4], our algorithm does not maintain the original structure of the tree. The network tree changes to relay different roles taken by nodes on the network. Since nodes will be transmitting, residual energy will vary for different nodes. Nodes which are more connected will exhaust their energy faster at first cycles. Due to dynamic dominating set construction, dominating nodes will switch roles to normal sensor nodes once their energy is below average

residual energy. Average residual energy is the average energy of all live nodes in the tile. Other nodes will assume the role of dominants and tile sinks. If nodes had mobility capability, dynamic nature of the network would depend on residual energy, neighbor node degree of nodes and relative speed at every time cycle as nodes move on the plane.

For a node $v \in V$ to be a dominant node v_{D_S} where $v_{D_S} \in D_S$:

- $\Delta \geq l$: Where l is a minimum defined number of neighbor (used as a threshold).
Example if l is set to 3 then the minimum defined number of neighbors is 3.
- $E \geq \text{min_}E$: Where $\text{min_}E$ is a minimum defined energy threshold. v_{D_S} can have $\Delta \leq l$ and $E \leq \text{min_}E$ if and only if it is on a path to form a connected dominating set tree and it is being used to form a connected dominating set.

For tile sink assignment in tile T_i (tile sink), v_{D_S} is a tile sink if and only if it has maximum values of Δ and E . When a node is a tile sink it is being added to the global dominator set of vertices V_{GD} , i.e. v_{D_S} is added to V_{GD} if and only if the values $(\Delta_{v_{D_S}}, E_{v_{D_S}})$ are $\max(\Delta_{D_S}, E_{D_S})$.

4.3 Dynamic Dominating Set Algorithm

We introduce a Dynamic Dominating set algorithm (DDS) which uses a connected dominating set explained in the previous section to construct a network graph which will be used for convergecast data collection. DDS constructs a tree graph by first finding the dominating set of the nodes in a tile. Dominating nodes are then joined to a connected tree. A dominator which will act as a tile sink is selected from among the dominating set. Tile sink participates in a global convergecast (convergecast between dominators of all tiles to the Sink).

A parent-child relation within nodes is created in this algorithm. This relation will be used in forwarding messages during the convergecast process. Then lastly a tile sink is selected and added to both the tile's dominating set (D) and Global dominators set (G_D). The tile sink is the node in set D with the maximum of residual energy and neighbor count.

The following variables are used in this algorithm:

- D : A set of nodes in a dominating tree in a tile
- G_D : A set of dominating nodes (dominators of all tiles as added to this set)
- Δ : Neighbor (one hop connection) count of a node
- E_v : Residual energy in a node in energy units
- $N(v)$: Set of neighbors of v
- P_v : Parent of node v
- $LeafSet$: A set of all nodes that are non-parent.
- $Temp_D, Temp_{D2}$: Temporary sets used to swap nodes during dominating set formation.

DDS runs in all nodes. DDS is a decentralized algorithm as per our assumptions in section 1.2. It starts by checking if nodes have enough energy and neighbor degree count to be added to dominating set. If so a node adds itself to a Dominant set D otherwise it adds itself to a non dominant set $LeafSet$. This sequence is repeated on neighbors of dominants ($N(v_D)$) and farther down the chain to the end where there are no more neighbors in range of all nodes in D . This is done in lines 1 to 18 in algorithm 4.1. Lines 19 to 21 connect nodes that have a dominant in range by creating edges. A tile sink is selected from set D and added to the global dominant

set G_D in lines 22 to 26. A parent relation is created in lines 28 - 33 between leaf nodes and dominants. Any leaf node selects one dominant in its range and assigns it a parent role to it by adding it to a parent set P_v .

A network tree created by DDS algorithm is a dynamic tree for convergecast in a WSN but can also be used for broadcast communication. The parent-child relationship can be used in a reverse direction to convey multi hop transmission in a broadcast communication.

4.4 Theoretical Analysis of DDS Algorithm

In this section, we prove that our algorithm will run, perform the required task and will terminate once task is completed or upon meeting a certain state. We also state that DDS algorithm has a time complexity of $O(N^2)$ in completing tree construction in one tile, where N is the number of sensor nodes on the network.

Lemma 4.4.1. *Algorithm 4.1 is correct and will terminate with a convergecast tree $T(k)$ constructed.*

Proof: Algorithm 4.1 takes in a graph network $G(V, E)$ (For each tile T_i) as input and generates a tree graph $T(k)$. From network model we know that there are k nodes in each tile. Proving by induction, assume V' is a set of nodes not in the tree yet. At first $V' = \{v_1 \dots v_k\}$ and sets $D = \{\emptyset\}$. A time $t = i$, set $V' = \{v_1 \dots v_{k-i}\}$ and set D will have $\{v_1 \dots v_i\}$, where D and *LeafNode* is a set of connected dominating sets and their leaf nodes which form a convergecast tree respectively. At any time $|V' + D| = |V| = k$. Therefore, this should also be true for $t = i + 1$ which sets V' and D will have $\{v_1 \dots v_{k-(i+1)}\}$ and $\{v_1 \dots v_{i+1}\}$ which also obeys $|V' + D| = |V|$. $\square$

Theorem 4.4.1. *Each tile forms a network tree in $O(N^2)$, where N is the number of sensors nodes on the network.*

Proof: DDS algorithm uses $k^2 + 5k + D + N$ steps to create a network tree and from parent relations where k , D and N are number of sensor nodes in a tile, number of tiles and number of sensors in the whole network respectively. Lines 1 to 18 uses k^2 steps to create a dominant tree. Dominant tree creation has a loop (FOR statements) which may run up to k steps which are nested in a LOOP(k) statement where k is the number of nodes in a tile. Global parent-child relationship formation (lines 28-33) takes N steps and tile sink appointment (lines 22- 26) takes D steps to go through all nodes in dominant set and appoint a tile sink. From $k^2 + 5k + D + N$ and $N = Mk$ we substitute the value of $k = \frac{N}{M}$ we get $\frac{N^2}{M^2} + 5NM + D + N$ where M is the number of tiles on the network which is a constant. Therefore DDS algorithm is bound to perform in $O(N^2)$. $\square$

Algorithm 4.1 DDS algorithm

```

1: for all  $v$  with  $(\Delta \geq l$  and  $E \geq \text{min\_}E)$  do
2:    $D \leftarrow v$ 
3: end for
4: repeat
5:   for all  $v \in N(v_D)$  do
6:      $\text{Temp\_}D2 \leftarrow v$ 
7:   end for
8:    $D \leftarrow \text{Temp\_}D$ 
9:    $\text{Temp\_}D \rightarrow \emptyset$ 
10:  for all  $v \in \text{Temp\_}D2$  do
11:    if  $\Delta \geq l$  and  $E \geq \text{min\_}E$  then
12:       $\text{Temp\_}D \leftarrow v$ 
13:    else
14:       $\text{LeafSet} \leftarrow v$ 
15:    end if
16:  end for
17:   $\text{Temp\_}D2 \rightarrow \emptyset$ 
18: until no node to add to  $D$ 
19: if  $v_D \in D$  are connected then
20:   Connect  $V_D \in D$ 
21: end if
22: for all  $v_D \in D$  do
23:   if  $(\Delta \geq \text{max\_}\Delta)$  and  $(E \geq \text{max\_}E)$  then
24:      $G_D \leftarrow v_D$ 
25:   end if
26: end for
27: {Global parent-child relationship formation}
28: for all  $v \in G$  do
29:   let  $P_V$ : Parent( $v$ )
30:   if  $v_D \in N(v)$  then
31:      $P_v \leftarrow v_D$ 
32:   end if
33: end for
34: Connect edges of  $\text{LeafSet}$ ,  $D$  and  $G_D$ 

```

Chapter 5

Local Collision Aware Convergecast Algorithm

In this chapter we present and describe our Local Collision Aware Convergecast Algorithm (LCAC). LCAC uses local collision model to control forwarding of messages in the network. LCAC takes a network tree constructed by DDS algorithm as input and uses the hierarchical relations between nodes to create a schedule to route information to the sink.

5.1 Convergecast

In our convergecast algorithm, an aggregate convergecast will be implemented by nodes forwarding their messages to their parents until the messages reach the sink. On every tile, convergecast starts when a tree is constructed and connected to the tile sink. When a node transmits or receives a message it uses its transceiver, hence energy is being used for that action. Every time a node transmits its energy is reduced by a tranceiving amount which will be specified on network setup.

We use a network graph generated by DDS algorithm to run our convergecast algorithm. A parent-child relationship generated by DDS algorithm is used to start

convergecast schedule from the leaves of the graph all the way up to the sink of the tile. We introduce an algorithm that has three layers of convergecast; from leaf nodes to dominant nodes, from dominant nodes to tile sink and from tile sink to global sink. The three phases ensure one way transmission as flow of information is from child to parent node, parent nodes to tile sink and from tile sinks to global sink.

Our algorithm obeys all convergecast properties listed in subsection 2.1.2 and proof of this will be seen in section 5.5. The next section details a local collision model which is obeyed by nodes on transmission to avoid collision in the network.

5.2 Local Collision Model

Collision in WSN is when a transmission is being interfered either on the sender's side or receiver's side and corrupts the message. Collision is more to individual nodes but the effect is for the whole network. We looked at collision as a local problem in a tile and in a node's neighborhood (transmission range) [21–23].

We introduce a local collision model, where transmitting nodes look at their local world and scan if there is any transmission in process to a node in the shared transmission zone. In a network of sensors where each node has a transmission zone of radius r , nodes close to one another will have overlapping transmission zones. In the overlapped area is where a receiver node can be situated to be able to have a successful transmission.

For collision to occur, say node A and B have an overlapped zone where node C is located. Collision will occur when both node A and B are sending to node C at the same time. With local collision model, a node checks if the recipient node is not receiving a transmission from another node before it sends its message. If there is a transmission in process, a node waits for the next round and tries again.

Local collision model is similar to common resource sharing solutions in programming and operating systems. In resource sharing a process or system must hold a token of the shared resource to be able to read, write or execute. Some of the common solutions to resource sharing problems in computer science are the use of semaphores [24], mutex [25] and the like. In our model, nodes use semaphore to hold a session of transmission so that nodes in the node's neighborhood won't interfere. There are two collision scenarios assumed. One is a parent having multiple transmissions from its children. The second one is interference between two adjacent transmissions.

1. Multiple sender's collision

Figure 5: Node A receives transmissions from node B and C in the same time slot.

Figure 5 shows nodes B and C have the same parent, node A. If it happens at some time slot i that both B and C transmit, a multiple sender collision will occur. To prevent this, a semaphore flag will be used within child-set of all nodes. For any node B or C to transmit, a semaphore variable *semaphore_A* must be held by the node transmitting. Once transmission is done it releases the semaphore. The other node will wait till the semaphore is released then it can hold the semaphore for transmission.

2. Interference between two transmissions

In figure 6, two adjacent transmissions (C to A and B to D) face collision due to interference between the two transmissions from two different pairs of nodes

Figure 6: Node A hears transmission from B while it is receiving a transmission from C.

when they are within each other's transmission range. A neighbor transmission semaphore is introduced to maintain who gets to transmit without collision at each time slot. For node *C* to transmit, it must hold a semaphore to prevent all other nodes around its neighborhood from transmitting. It must also check that no node in the neighborhood it is in is transmitting.

Local collision model assumes collision in payload messages only. It does not take consideration control messages of the network. This is because to implement a semaphores in WSN will require alot of control messages between nodes so as a node can hold a semaphore. We assume that control messages can be transmitted in a broadcast manner so they will be heard by all nodes and only the target nodes acknowledge to the message.

5.3 LCAC Algorithm

In this section we present a Local Collision Aware Convergecast algorithm (LCAC) that runs on nodes which are in a DDS network graph. LCAC algorithm performs an aggregate convergecast to a network of sensor nodes which are in a hierarchical structure with the sink node on the highest level. A node forwards a message to a linked node on a higher level. The message is merged and forwarded all the way up the hierarchy to the sink. The main reason to use use aggregate convergecast

is to minimize schedule time, save energy and reduce network traffic by having less transmission in the network.

LCAC implements Local collision model by having semaphores [24] that a transmitting node holds until done with transmission. Only a node with all the semaphores is allowed to transmit. There are two semaphore variables in any local neighborhood of a node; A *parent-semaphore* and a *neighbor-semaphore*. A parent-semaphore is a session variable for nodes under the same parent node while neighbor-semaphore is a session variable for all the nodes in a neighborhood zone of a node. Each node checks these variables before transmitting in the *Send_message()* procedure.

Convergecast process starts at the leaves of the graph (nodes in *LeafSet*) in lines 2 - 11 in algorithm 5.1. Transmitting nodes are chosen at random. For nodes in *LeafSet* who transmitted, they mark their transmission flag as *true* so they won't transmit again in that round. And those who didn't transmit they mark with a *false* so the parent node can know that it attempted but failed or does not have sufficient energy to transmit.

In lines 13 to 21 dominants in every tile check if all leaf nodes transmitted or attempted transmission before they merge their message with the received messages and forwards to their higher dominants or tile sink. Tile sink then forwards the collected message in one transmission to the global sink in lines 23-38. Convergecast in dominants and tile sink follow the same procedure of checking for lower level acknowledgment on transmission. Tile sink transmits with collision aware mode and then updates transmission flag.

The following variables are used in the convergecast algorithm;

- Transmit-flag(k): A transmission flag, a boolean array for all nodes where k is a node number. Initially transmit-flag (k) = NULL

- `Send_message()`: A send message procedure presented by algorithm 5.2, forwards a message to the specified node
- P_v : Parent of node v
- `Energy_per_transmission`: Energy value which is deducted from a node's residual energy when it transmits.

5.4 System Flow

As said earlier, our network system is decentralized. Our two algorithms will be run by each node in the network individually. Nodes will sense each other and organize themselves into a network. Nodes will scan the network and forward messages to the sink until majority of nodes exhaust all energy that a convergecast tree cannot be formed. Algorithm 5.3 gives a system flow that each node will run on the network.

5.5 Theoretical Analysis of LCAC Algorithm

In this section, we prove that our algorithms will run, perform the required task and will terminate once task completed or upon meeting a certain state. We also present runtime complexity for LCAC to conduct convergecast in three layers of the network. These layers are from leaf nodes to dominants, from dominants to tile sinks and from Tile sinks to global sink.

Lemma 5.5.1. *Algorithm 5.1 will complete one convergecast round and terminate with all data forwarded to sink.*

Proof: Algorithm 5.1 performs a one way message forwarding convergecast from leaf nodes, to the tile sink and finally to the global sink. At any time slot i , parent

nodes do not transmit until their children sets have forwarded their message. A variable *Transmission-flag* (v) is used to control transmission. When a node transmits, it marks its transmission flag variable as *True*. If it could not transmit due to a collision it marks as *False*. A parent node will transmit only if all its children nodes have either *True* or *False* (Transmission-flag is initialized as *NULL*). Having flags in LCAC algorithm limits infinite transmission of nodes.

Theorem 5.5.1. *One complete convergecast from leaf node to the sink is done in $O(N)$, where N is the number of sensor nodes on the network.*

Proof: In the LCAC algorithm, convergecast in leaf nodes (lines 2-11) takes $(8(k - D))$ steps where k is the number of nodes in a tile and D is the number of dominant nodes in the tile. Convergecast in a *LeafSet* has a total of 8 steps are run by $k - D$ nodes in a tile. In calculating the run time of this algorithm, we also include the steps of `Send_message()` procedure as seen in algorithm 5.2 which are 5 steps.

Convergecast in dominating set takes $7D$ steps (lines 13-21). Each tile has D dominating nodes. Data collection at this layer takes a maximum of 7 steps for all D nodes.

Global convergecast of tile dominators (Lines 23-38) to the sink takes $8M$ steps, where M is the number of dominators which is the number of tiles on the plane. This results to a linear function of

$$7k - D + 8M.$$

From the equation $N = Mk$ in section 4.1 we get $k = \frac{N}{M}$. Substituting the value of k to our algorithm's equation we get

$$7N - MD + 8M^2$$

□

Lemma 5.5.2. *A node obeying local collision model will transmit at a collision free media only if neighbor nodes obey local collision model.*

Proof: At time t_i node v scans its neighborhood $N(v)$ before it transmits to ensure the media is free. It also ensures that its parent $P(v)$ is not receiving a transmission from another node. A node must hold two semaphores to be able to transmit as seen in algorithm 5.2. The two semaphores are *local_semaphore* and *parent_semaphore*. As explained in section 5.2, only a node that has both semaphores can transmit otherwise they wait for another round. $\square$

Algorithm 5.1 Convergecast

```

1: {Convergecast process starts in the leaves of graph ( $v \in Leafnodes$ ) }
2: for all  $v \in LeafSet$  do
3:   Let  $do\_convergecast = Random(1)$  {A random number less or equal to one is
   generated}
4:   if  $do\_convergecast \geq Transmission\_factor$  then
5:     Send_message( $P_v$ )
6:      $E \leftarrow E - Energy\_per\_transmission$ 
7:      $Transmission\_flag(v) \rightarrow true$ 
8:   else
9:      $Transmission\_flag(v) \rightarrow false$ 
10:  end if
11: end for
12: { Convergecast of CDS nodes }
13: for all  $v_D \in D$  do
14:   if  $N(v_D)$  have  $Transmission\_flag = trueorFalse$  then
15:     Send_message( $P_v$ )
16:      $E \leftarrow E - Energy\_per\_transmission$ 
17:      $Transmission\_flag(v_D) \rightarrow true$ 
18:   else
19:      $Transmission\_flag(v_D) \rightarrow false$ 
20:   end if
21: end for
22: { Global Convergecast }
23: for all  $v_{GD} \in G_D$  do
24:   {Checks if a node has no child}
25:   if  $N(v_{GD}) = \emptyset$  then
26:     Send_message( $P_v$ )
27:      $E \leftarrow E - Energy\_per\_transmission$ 
28:      $Transmission\_flag(v_{GD}) \rightarrow true$ 
29:   else
30:     {Checks if Children nodes have transmitted}
31:     if  $N(v_{GD})$  have  $Transmission - flag = true$  then
32:       Send_message( $P_v$ )
33:        $E \leftarrow E - Energy\_per\_transmission$ 
34:        $Transmission\_flag(v_{GD}) \rightarrow true$ 
35:     end if
36:      $Transmission\_flag(v_{GD}) \rightarrow false$ 
37:   end if
38: end for

```

Algorithm 5.2 Send_message(Parent P)

```

1: {Checks if Parent and neighbor semaphores are free}
2: if local_semaphore = False and parent_semaphore(P) = False then
3:   {Semaphores are locked so no other node can hold them while one is transmitting}
4:   local_semaphore  $\rightarrow$  True
5:   parent_semaphore(P)  $\rightarrow$  True
6:   forward message to parent P
7:   {Freeing semaphores}
8:   local_semaphore  $\rightarrow$  False
9:   parent_semaphore(P)  $\rightarrow$  False
10: else
11:   Wait for next round
12: end if

```

Algorithm 5.3 System flow

```

1: Initialize (scan neighborhood)
2: repeat
3:   Broadcast E and  $\Delta$  to neighbors
4:   DDS algorithm {Take part in constructing a network}
5:   LCAC algorithm {Data collection phase}
6:   Update (E,  $\Delta$ )
7: until no tree is formed (Based on nodes alive)

```

Chapter 6

Analysis of Algorithms

In this chapter we present theoretical and simulation analysis of our algorithms. In theoretical analysis we prove that our network life time depends on average energy of all nodes on the network. We also prove that LCAC algorithm obeys all the properties of convergecast. In simulation analysis, we give a description of simulation setups, different tests conducted on our convergecast model and give a detailed simulation analysis of our model.

6.1 Theoretical Analysis

In this section we present theoretical analysis of our network by stating and proving that our overall network's time complexity is dependent on the number of nodes alive in the network. We will also prove that LCAC algorithm adheres to convergecast properties.

Theorem 6.1.1. *The network (algorithm 5.3) will run in a minimum of z convergecast rounds, where*

$$z = \frac{\sum_{i=1}^N \text{Energy_}E_{v_i}}{\text{Energy_per_transmission}}$$

Proof: Assuming all sensor nodes on the network have messages to send to the sink in every convergecast round and the network is collision free. For any sensor node to transmit, the probability value of Do-convergecast $\geq$ Transmission-factor where *Do-convergecast* is a random number variable $[0,1]$ which if greater than *Transmission-factor* then a node transmits. *Transmission-factor* is a user specified probabilistic value which helps to message packet generation on the network.

By Lemma 4.4.1 and 5.5.1, on every convergecast round a network tree will be created successfully and nodes perform convergecast schedule. On every convergecast round each node with transmit once. So for a network of N sensor nodes it would take a network $N \frac{Energy_{E_{v_i}}}{Energy_{per_transmission}}$ convergecast rounds till when nodes run out of energy assuming also that $min_E = 0$, where $Energy_{E_{v_i}}$ is the residual energy of node v_i . $\square$

We state and prove that our convergecast algorithms obeys all convergecast properties stated in section 2.1.2.

Theorem 6.1.2. *Algorithm 5.1 obeys all properties of convergecast transmission in WSN.*

Proof: LCAC algorithm obeys all the properties of a convergecast schedule as listed in subsection 2.1.2.

Convergence transmission: LCAC runs on a tree constructed by DDS algorithm which is in a reverse funnel structure. The network tree is rooted at the tile sink and spans all the way to the leaf nodes. Nodes forward their message to the parent node only, the parent node is a node in a higher hierarchy than itself. Tile sink being the highest node in the hierarchy.

Single transmission per node: LCAC has transmit-flag variable which acts as an acknowledgment that a node has transmitted so it will not take part or waited for by its parent node in that round of convergecast.

One way transmission: A node v can forward a message to its parent ($P(v)$) only.

Collision free transmission: Because of Lemma 5.5.2 any node transmitting in a network of nodes running LCAC, is transmitting in a collision free media. $\square$

6.2 Simulations

6.2.1 Simulation Setup

We simulated our algorithms in Netlogo environment. Netlogo is a programmable modeling software for simulating social and scientific systems [26]. Netlogo works best with artificial and self organising systems since it has mobility and multi agent programming capability. Our network model was not extended from any of the models in model library or those developed by different people in Netlogo's online community [27].

Figure 7: Snapshot of WSN model developed in Netlogo

Our network model has user input interfaces for different variables and parameters

for use. As seen in figure 7, our model has parameter input and model launch panel and model world where network simulation is displayed. It also has a data presentation panel on the right where different graphs are being plotted as the simulation runs. The control panel has three actions buttons:

- SETUP which sets up the model by creating the tiles on a plane and randomly creating and distributing sensor nodes on the tiled plane.
- SIMULATE runs the system, which is shown in algorithm 5.3 continuously until stopped or until the overall network dies.
- SIMULATE ONCE! runs only one cycle of the system when pressed.

Figure 8: WSN model's control panel

The rest of user input interfaces as shown on figure 8 are explained as follows:

1. **tile_size**; A slider that specifies the length of each square tile in distance units. One unit equals to one cell which is the smallest division on the plane which can contain only one node.
2. **nodes-per-tile**; A slider that sets how many sensor nodes can be deployed in one tile. If the number is more than a tile's maximum number of cells then it only deploys the maximum number of nodes. For example, tiles with `tile_size`

= 20 can have up to $tile_size^2$ nodes which is 400 nodes per tile. So if nodes-per-tile was set to 500 then only 400 nodes will be deployed.

3. **Neighbor-degree-distance**; This is a radius of a transmission zone of a node. It is applied to all nodes in the tile. Again a distance unit equals to a length of a cell on the plane. Figure 9 demonstrates a nodes transmission zone of different values of this parameter.
4. **Initial_Energy**; An input field to which a user enters how much energy units a node should be deployed with. This is applied to all nodes on the network.
5. **Energy_per_transmission**; This input field sets the cost of transmission per node of a network. Any node transmitting or receiving a message on the network uses this amount of energy to do so.
6. **Dominant_energy_Threshold**; A minimum energy unit required for a node to be a dominant.
7. **Alive_Energy_Threshold**; A minimum energy unit a node required for a node to take part in convergecast schedule, otherwise its marked as a dead node.
8. **Min_Delta**; A slider to specify the minimum neighbor counts required by a node qualify to join tile's dominant set as explained in section 4.2.
9. **Transmission-factor**; A probabilistic [0 1] value which controls percentage volume of nodes transmitting on one cycle.
10. **Dominant_per_neighborhood**; This slider sets a limit of how many dominants to be selected based on the setting of the network. This is used to generate minimum and efficient dominant networks.

Figure 9: Netlogo's world showing a blue node at the center with its transmission zone in Grey for values of Neighbor-degree-distance at: (a) 2 (b) 3 (c) 4 (d) 5 (e) 6

Figure 10 demonstrates a running network model at 290th cycle. Different roles of sensor nodes in the networks are color coded for visual clarity. Leaf node sensors are Yellow. Dominants are Blue. Tile sinks are Orange and tiles are shown by White and Grey backgrounds. A global sink is colored Green and situated at coordinates (0,0) which is at the center of plane but it is not displayed most times the network runs due to its larger size compared to other nodes and the fast graphics changes.

A tick in Netlogo is a timer function which counts how many times a procedure has run. For our case it counts the RUN procedure which on every cycle it calls DDS algorithm to reconstruct the graph then calls LCAC algorithm to perform converge-cast.

Figure 10: A snapshot of a running model showing how nodes take different roles on every cycle.

6.2.2 Performance Metrics

In our tests we will run our network model on different settings of parameters so we can capture the reaction of the network at different scenarios. Based on our design objectives, we focused on the following performance metrics when testing our

algorithm:

- **Time of execution**, we test how fast our algorithms can perform on different network settings. Different network settings produce different effect to the convergecast process. These changes would also impact the execution time of an algorithm.
- **Energy conservation**, we monitor how our model consumes energy and at what setting of the network our model will be energy efficient. This metric will assess how our network's energy consumption trend changes with different network settings.
- **Collision avoidance**, how different settings of the network affect collision in the network and how our algorithms perform to reduce it.
- **Mass coverage**, how many nodes transmit and how many dominants transmit in every cycle. This determines how our model can perform either in real time high availability systems.

We will compare our results with that of WIRES (Weighted Incremental Ranking for convergEcast with aggregation Scheduling) algorithm [4]. WIRES design is very similar to our design. It uses neighbor degree as a weighing parameter. We implement WIRES on Netlogo and run it on a network generated by our algorithm (DDS) which is also similar to the tree construction algorithm in [4]. All of our tests were run on a computer with a AMD Athlon II X4 645 Processor with 3.10GHz processor speed and a 4GB RAM.

6.2.3 Initial Test Results

Our initial test was run with LCAC algorithm as the results were compared with those of WIRES. WIRES uses TDMA and weighted neighbor count of nodes to prioritize

which node should transmit first while LCAC uses a hierarchy tree structure from DDS algorithm which has dominants and tiles sinks to conduct convergecast. This initial test was to evaluate how our algorithm which does not use any TDMA scheme performs compared to one which uses TDMA. The network model for this test was a grid network with a *neighbor-degree-distance* set to 2. Hence a node's neighborhood is a one-hop world.

We setup three test cases which were all based on evaluating the effect of node density on the model's performance. We called the three tests Low density, Medium density and high density. Table 1 gives the parameter setting of each of the test case. Initial_Energy, Energy_per_transmission and average convergecast rounds were constantly at 1000 units, 30 units and 200 rounds respectively.

Test case	Nodes-per-tile	Number of tiles	min_Delta
Low density	200	4	2
Medium density	300	4	2
High density	400	4	3

Table 1: Parameter settings for initial test cases

Results of these tests are presented with three types of graphs plotting values from test results of both algorithms. These graphs are: graph of average residual energy (average energy remaining per node) with time, average number of nodes alive with time and average number of nodes transmitted with time. The graphs we present are an average of 10 runs each of 200 rounds of convergecast with a confidence level of 95%. For all of our results, we refer to time as number of cycles or rounds an algorithm runs.

(a)

(b)

(c)

Figure 11: Low density test- Graphs of (a) Average residual energy (b) Average nodes alive and (c) average Number of nodes transmitted with time

Figure 12: Medium density test:- Graphs of (a) Average residual energy (b) Average nodes alive and (c) average Number of nodes transmitted with time

(a)

(b)

(c)

Figure 13: High density test:- Graphs of (a) Average residual energy (b) Average nodes alive and (c) average Number of nodes transmitted with time

Graphs on figure 11 present results of low density test for both WIRES and LCAC. Figure 12 displays graphs of medium density test results and finally figure 13 presents graphs for high density test results.

By Low density we refer to minimum number of nodes per unit space, for our case we measure it by varying number of nodes in a tile. We used 200 nodes per tile for a tile that can have up to 400 nodes per tile. Medium density refers to an intermediate number of nodes per tile. We used 300 nodes per tile for this test. For high density we refer to having a maximum number of nodes in a unit space of the plane. We used 400 nodes per tile which implies that each cell in the plane had a sensor node.

We performed the three tests for different node density on the plain so we can evaluate how the network behaves when the number of nodes varies. We discuss the effect of varying node density of the network in energy conservation, convergecast schedule time, collision on the network and number of nodes taking part in convergecast schedule.

Analysis from plotted graphs on figures 11,12 and 13 showed LCAC exhaust more nodes energy compared to WIRES. This is because in every convergecast round, LCAC algorithm has more nodes transmitting compared to WIRES algorithms. LCAC algorithm has more nodes transmitting during convergecast because all nodes check on local traffic only. With a grid network and a *neighbor-degree-distance* of 2 units, nodes are limited to see all the nodes in their neighbor range.

Netlogo has a neighbor function for nodes in the simulation model called **Neighbors** [26]. **Neighbors** calls the 6 surrounding neighbor nodes which are in a one-hop range. So if a node has more than 6 neighbor nodes in its range, only six of them are considered as the neighbor set of a node during convergecast data collection. Therefore nodes in the network recognize fewer neighbor nodes than they are suppose to hence they transmit and make their decisions based on those few recognized nodes. In this test, the local collision model in LCAC algorithm was not accurate for higher

density networks. Hence in all test cases WIRES was more energy efficient compared to LCAC.

Graphs on figures 11,12 and 13 shows how both models maintain number of nodes on the network. WIRES proved to be more optimal by maintaining a steady and high number of nodes because of its energy efficiency due to TDMA.

Figures 11,12 and 13 present graphs of average number of nodes transmitted when both models were ran in the initial tests. LCAC has more network coverage by having more number of nodes transmitting compared to WIRES. This is because of its dynamic and locality nature of each node to scan their local neighborhood only compared to how WIRES checks the whole network in its TDMA module of WIRES which limits the number of nodes to transmit per time.

For medium and high node density, LCAC had high network coverage for a short network lifetime before the overall network ran out of power. The advantage of this is LCAC covers more area in a short time by providing sensed information to the global sink. LCAC is more reactive compared to WIRES. But LCAC has a disadvantage of using more power compared to WIRES. In the next subsection, we will implement TDMA in LCAC then test the two models again in a non grid network using the performance metrics described in subsection 6.2.2.

6.2.4 Test Results with TDMA

In subsection 6.2.3 we presented results of initial tests done on our algorithms. LCAC didn't perform well compared to WIRES because of high number of nodes transmitted per time hence the overall network life time was minimized. Our objective is to have a network model which optimizes minimal schedule time, energy efficiency and collision minimization. We implement TDMA on LCAC algorithm and test it with WIRES to further evaluate its performance. We implemented a non grid network by setting a slider for *Neighbor-degree-distance* variable so a user can set how many distance units

to use.

We performed three test sets where in each set we varied one of the parameter of our network model. The three test sets are Node density test, Neighbor degree test and Time test. We will present the test cases and test results of each test set as compared to results of WIRES [4]. Each plotted graph is an average of 5 runs and each run has at least 200 rounds of convergecast with a confidence interval of 95%. Our results are present in four graphs per test. We plot a graph of average residual energy, average nodes alive, average nodes transmitted and average nodes on collision with time. For LCAC the number of nodes on collision plotted are nodes which avoided collision as seen in algorithm 5.2, a node transmits only if it has both semaphores otherwise it waits.

Node density test

In Node density test, we are evaluating the performance of our algorithm when node density is varied. The two models were tested on the effect of increasing number of nodes on the plane. Tests were done for {100, 150, 200, 250} values of number of nodes per tile. Table 2 shows parameter setting for the tests. Other parameters were kept constant for the entire 4 tests. The values of constant parameters are Neighbor-degree-distance = 4, Initial_energy = 1000 units, energy_per_transmission = 30 units and min_Delta = 5.

Test	1	2	3	4
Number of nodes	100	150	200	250

Table 2: Parameter settings for Node density tests

(a)

(b)

(c)

(d)

Figure 14: Node density Test 1- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

(a)

(b)

(c)

(d)

Figure 15: Node density Test 2- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

(a)

(b)

(c)

(d)

Figure 16: Node density Test 3- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

(a)

(b)

(c)

(d)

Figure 17: Node density Test 4- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

We present the results of this test on graphs in figures 14, 15, 16 and 17. A detailed analysis and discussion of these results will follow on section 6.3.

Neighbor degree test

These tests were done by varying the node's neighbor distance range to assume values $\{2,3,4,5,6\}$ distance units. Figure 9 demonstrates neighbor distance range of a node on different distance units. Constant parameters in this test are Node-per-tile = 200, Initial_energy = 1000 units, energy_per_transmission = 30 units and min_Delta = 5. Table 3 gives parameter setting for this test case. min_Delta is proportional to neighbor degree distance (radius) hence it is also changed during the test.

Test	1	2	3	4	5
Neighbor degree distance	2	3	4	5	6
min_Delta	4	4	4	6	6

Table 3: Parameter settings for Node degree tests

Graphs on figure 18,19, 20, 21 and 22 present results of this test on graphs. A detailed analysis and discussion of these results will follow on section 6.3.

(a)

(b)

(c)

(d)

Figure 18: Node degree Test 1- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

(a)

(b)

(c)

(d)

Figure 19: Node degree Test 2- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

(a)

(b)

(c)

(d)

Figure 20: Node degree Test 3- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

(a)

(b)

(c)

(d)

Figure 21: Node degree Test 4- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

(a)

(b)

(c)

(d)

Figure 22: Node degree Test 5- Graph of (a) Average Residual energy (b) Average nodes alive (c) Average transmitting nodes and (d) Nodes on collision

Time test

We conducted performance time test for LCAC and WIRES to determine how much time each algorithm uses to execute one round of convergecast in Netlogo. We had two cases in this test. We evaluated algorithms time performance when varying node density and neighbor degree radius. We used the same parameter setting displayed in figure 2 and 3 to conduct time test for node density variation and neighbor degree radius variation respectively. Results of this test are an average of five runs with different random nodes deployed in the plane where in each test. The algorithms had 30 calls (rounds).

Figure 23 and 24 present results of execution time test in graphs of time versus number of nodes and time versus neighbor degree radius respectively. In the next section we will give a detailed analysis and discussion of results from these tests.

Figure 23: Graph of time (ms) vs. Number of nodes

Figure 24: Graph of time (ms) vs. Neighbor degree radius (unit)

6.3 Performance Evaluation

We present an evaluation discussion on our test results based on performance metrics described in section 6.2.2. Our results from section 6.2.4 are presented in four graphs in each test. The graphs are averages of residual energy, number of nodes alive, number of nodes transmitted and number of nodes on collision. We saw initial results of these algorithms in section 6.2.3 where LCAC did not perform well in energy conservation, maximizing network life time and maximize mass network coverage. In this section we will discuss performance of LCAC compared to WIREs when they are both using TDMA sequence using results presented in section 6.2.4.

6.3.1 Time of Execution

Figure 23 and 24 show graphs of time test results by varying node density and neighbor degree range respectively in both LCAC and WIREs. LCAC proved to complete a round of convergecast faster than WIREs in denser network than less dense networks.

This can be seen on figure 23. LCAC used less time in 250 and 300 nodes per tile while it used significantly more time to complete a round of convergecast in 150 and 200 nodes per tile. Its time efficiency can be associated with its aggregate and dynamic nature, whereby a network tree is composed of nodes (Dominants) which are more connected to their neighbors hence uses a shorter path to reach the sink.

At local level, LCAC also proved to be time efficient in dense networks. Figure 24 shows graph of time taken by LCAC and WIRES to complete one round of convergecast in networks with different neighbor distance range. LCAC had minimum running time in 4, 5 and 6. Neighbor distance is proportional to local traffic and connectivity at node level hence LCAC is more time efficient at dense neighborhood of a node.

6.3.2 Energy Conservation

Initial test results in section 6.2.3 showed that LCAC exhausted more energy than WIRES and LCAC had minimum network life time. That was because of improper transmitting sequence of nodes in LCAC. Nodes only scanned their local world for collision without having a time sequence of transmission. In section 6.2.4 we presented tests for both models with TDMA sequence and presented results for node's average residual energy for both test cases. We know that total energy consumption of the network is proportional to the number of nodes taking part in convergecast. Hence energy efficiency of both models depends on how many nodes take part in convergecast and how many had a successful transmission in the network.

For Node density, LCAC maintained the number of nodes transmitting in all values of *nodes-per-tile*. Figures 14, 15, 16 and 17 shows number of nodes transmitted (plot (c)) in LCAC to an average of 10 nodes while that of WIRES decreases in a linear manner as the network progresses over time. Because of its near constant number of nodes transmitting, LCAC had a stable energy consumption slope (plot (a)) compared

to WIRES which had a lower slope. This implies that WIRES had less overall network energy retained compared to LCAC.

In node degree test, we see the effect of high network traffic due to nodes having more neighbor connectivity as values of *Neighbor-degree-distance* increases in both models. Figures 18, 19, 20, 21 and 22 presents results of node degree tests. LCAC had decreasing number of nodes transmitting (plot (c)) as values of *Neighbor-degree-distance* increases. This is due to high collision rates as neighbor connectivity increases and therefore local collision model forces nodes to wait if collision is sensed. WIRES is seen to have the same linear decreasing pattern as in Node density test. Plot (a) displays graph of average residual energy with time. LCAC showed a stable decreasing slope compared to WIRES due to local collision model taking effect.

Comparing energy consumptions of node density test and node degree test for both algorithms we see that with increasing neighbor connectivity (node degree distance) a network model consumes more energy compared to increasing nodes in the network with less neighbor connectivity.

6.3.3 Collision Avoidance and Minimization

In section 6.2.4, we evaluated a TDMA version of LCAC with WIRES which has TDMA as its sequential and collision control mechanism. Time Division Multiple Access (TDMA) is transmission control mechanism where different nodes are assigned different time slots so that nodes which are close by each other or within each other's transmission range. Nodes transmit and receive messages at different time slots to avoid collision. We introduced local collision model in our convergecast algorithm which is a collision sense and avoidance model. Having TDMA and local collision model in one algorithm proved to improve quality of service by preventing collision and minimizing flooding of messages on the network. We see this in results of both tests; plot (d) of test results figures show the average number of nodes transmitted

with collision for WIRES and average number of nodes sensed and avoided collision on LCAC. LCAC has more nodes waiting (avoiding collision) to transmit than number of nodes transmitted. This proves that having dual collision control mechanism filters collision although it limits instant flow of information to the sink. Hence LCAC is not suitable for high availability real time systems.

6.3.4 Mass Network Coverage

An ideal WSN network is supposed to sense and submit information of the whole network to the sink in a short time. Sink should be able to recreate the whole network from messages routed to it. Mass network coverage can be assessed by how many number of nodes transmit in one round of convergecast and network's life time, assuming that all nodes generate or sense information equally. From test results in section 6.2.4, LCAC shows high network life time compared to WIRES in plot (b) by having more nodes alive on the network as the model progresses. But with low number of nodes transmitting per round, LCAC takes more time to collect information of the network to the sink when compared with WIRES. WIRES has high network coverage compared to LCAC, only that WIRES dies fast and has key nodes (dominant) exhausted more because of its static nature.

Chapter 7

Conclusion and Future Work

We defined a convergecast problem as a joint objective problem of minimizing convergecast time, optimizing energy efficiency and minimizing collision on the network. We introduced two algorithms, DDS in Chapter 4 and LCAC algorithm in Chapter 5. DDS algorithm creates a network tree of sensor nodes using dominating sets. LCAC algorithm uses local collision model to route messages to the sink using the tree infrastructure from DDS respectively.

DDS algorithm uses node's residual energy and neighbor count as metrics for dominating set selection in a dynamic manner. This means that the WSN system runs and refreshes the tree in every convergecast round. Thus maintaining nodes with higher energy and neighbor count as dominant to sustain traffic of other sensing nodes.

LCAC algorithm uses a tree constructed by DDS algorithm to conduct convergecast to collect sensed data to sink. LCAC obeys convergecast properties in WSN. LCAC uses local collision model to minimize collision by collision avoidance. Local collision model described in section 5.2 uses node's local traffic information to regulate when a node can transmit.

We present theoretical proofs of DDS and LCAC algorithm in sections 4.4 and 5.5 respectively. We proved that both algorithms are correct. We proved that DDS

algorithm has a runtime complexity of $O(N^2)$ to construct a network tree in one tile where N is the number of nodes on the network. We proved that LCAC algorithm performs one convergecast round in $O(N)$. We also proved that our network model's life time is dependent on the number of nodes alive in the network and that LCAC algorithm obeys all properties of convergecast in section 6.1.

We simulated our network model on Netlogo. We conducted several tests and compared the results with a similar network model which runs WIRES [4]. WIRES uses neighbor count of nodes as weights to prioritize message transmission on the network. Our initial test results showed that LCAC was outperformed by WIRES because LCAC was not using TDMA like WIRES.

Our final test results in section 6.2.4 where both LCAC and WIRES had TDMA implemented showed great results for LCAC where it outperformed WIRES in schedule time minimization, energy efficiency and collision minimization. Final results also showed that LCAC has less network coverage in terms of number of nodes transmitting in a convergecast. LCAC algorithm is not suitable for sensitive and high availability systems because of low network coverage due to implementation of the local collision avoidance model.

7.1 Future Work

Much effort is put on improving convergecast and data collection in WSN. Many possibilities, assumptions and approaches have been used to bring the level in creation and innovation to the current state. Advancement in technology also accelerates inventions of better WSN. We have not yet met the hardware compatibility in optimizing wireless sensor unit hardware. More work has to be done so as to come up with an optimal and efficient convergecast schedule.

Data collection in wireless networks is being affected by several factors. Some of

the factors are latency, bandwidth, energy consumption and mobility support and fault tolerance. An ideal WSN is one which considered to balanced all these factors. In reality this is very hard to achieve though with certain trade-offs. One can integrate the most out of these features. In this work we mainly addressed convergecast schedule minimization time, energy efficiency and minimization of collision on the network. We introduced a joint objective WSN model. More features have to be integrated to our work to reach an ideal WSN model and then even more work has to be done to realize and perfect such a model. Some of the future work to be done to improve our work are:

1. Real implementations of research works should be one of the future works to be considered. A lot of works being done on WSN are being tested and compared on simulations over several test beds while in real life the environments are different and may affect the WSN system in some way. Most schedules are proposed on a protocol model while few are being advanced to a SINR model. To run a schedule on real sensor network is lot of work and expensive but it gives a true performance and behavior of the schedule in real life.
2. Adding mobility (sensor nodes mobility or sink mobility) support to our algorithms is one of the future works to extend and enable selection of tile sinks and dominants based on relative velocity of sensor nodes. Applying mobility nodes would facilitate a more dynamic network and can be used to model sensor networks in VANETS (Vehicular Ad-hoc Networks).
3. Adding smart routing technique to nodes which will enable them to have more capability of having sensor nodes select a closer parent (dominant) even if it is not on the same tile (region). Currently our DDS algorithm binds a node to the network tile it is located at. So a node must connect to the tile's network tree only. Smart routing will form an effective load balance between dominants

and tile sinks and thus have more dominants up boosting network life time.

List of References

- [1] C. Townsend and S. Arms. “Chapter 22: Introduction to wireless sensor networks.” In “Wireless Sensor Networks: Principles and Applications,” pages 439–450 (2004).
- [2] S.-H. Huang, P.-J. Wan, C. Vu, Y. Li, and F. Yao. “Nearly constant approximation for data aggregation scheduling in wireless sensor networks.” In “INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE,” pages 366 –372. ISSN 0743-166X (2007).
- [3] X. Chen, X. Hu, and J. Zhu. “Minimum data aggregation time problem in wireless sensor networks.” In X. Jia, J. Wu, and Y. He, editors, “Mobile Ad-hoc and Sensor Networks,” volume 3794 of *Lecture Notes in Computer Science*, pages 133–142. Springer Berlin / Heidelberg. ISBN 978-3-540-30856-0. 10.1007/11599463_14 (2005).
- [4] B. Malhotra, I. Nikolaidis, and M. A. Nascimento. “Aggregation convergecast scheduling in wireless sensor networks.” *Wirel. Netw.* **17**, 319–335. ISSN 1022-0038 (2011).
- [5] N. Karthickraja, V. Sumathy, and M. Jabeer Ahamed. “A novel hybrid routing protocol for data aggregation in agricultural applications.” In “Communication Control and Computing Technologies (ICCCCT), 2010 IEEE International Conference on,” pages 227 –231 (2010).
- [6] B. N. Clark, C. J. Colbourn, and D. S. Johnson. “Unit disk graphs.” *Discrete Mathematics* **86**(13), 165 – 177. ISSN 0012-365X (1990).
- [7] J. Gao and L. Zhang. “Well-separated pair decomposition for the unit-disk graph metric and its applications.” In “STOC,” pages 483–492 (2003).
- [8] Wikipedia. “Memory hierarchy — wikipedia, the free encyclopedia.” [Online; accessed 12-Aug-2012] (2012).

- [9] V. Annamalai, S. K. S. Gupta, and L. Schwiebert. "On tree-based convergecasting in wireless sensor networks." In "Proceedings of the IEEE Wilress Communications and Networking Conference WCNC03," (2003).
- [10] S. Upadhyayula, V. Annamalai, and S. Gupta. "A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks." In "Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE," volume 6, pages 3525 – 3530 vol.6 (2003).
- [11] E. Lee, S. Park, F. Yu, and S.-H. Kim. "Communication model and protocol based on multiple static sinks for supporting mobile users in wireless sensor networks." *Consumer Electronics, IEEE Transactions on* **56**(3), 1652 –1660. ISSN 0098-3063 (2010).
- [12] C. Kam and C. Schurgers. "Converss: A hybrid mac/routing solution for small-scale, convergecast wireless networks." *Mobile Computing, IEEE Transactions on* **10**(9), 1227 –1236. ISSN 1536-1233 (2011).
- [13] S. Gandham, Y. Zhang, and Q. Huang. "Distributed minimal time convergecast scheduling in wireless sensor networks." In "Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on," page 50. ISSN 1063-6927 (2006).
- [14] Y. Yuan and Y. Peng. "Racetrack: An approximation algorithm for the mobile sink routing problem." In I. Nikolaidis and K. Wu, editors, "Ad-Hoc, Mobile and Wireless Networks," volume 6288 of *Lecture Notes in Computer Science*, pages 135–148. Springer Berlin / Heidelberg. ISBN 978-3-642-14784-5. 10.1007/978-3-642-14785-2_11 (2010).
- [15] K. Veena and B. Kumar. "Convergecast in wireless sensor networks: A neural network approach." In "Internet Multimedia Services Architecture and Application(IMSAA), 2010 IEEE 4th International Conference on," pages 1 –6 (2010).
- [16] Y. Zhang and Q. Huang. "Coordinated convergecast in wireless sensor networks." In "Military Communications Conference, 2005. MILCOM 2005. IEEE," pages 1152 –1158 Vol. 2 (2005).
- [17] A. Kesselman and D. Kowalski. "Fast distributed algorithm for convergecast in ad hoc geometric radio networks." In "Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on," pages 119 – 124 (2005).

- [18] A. T. Erman, T. Mutter, L. van Hoesel, and P. J. M. Havinga. “A cross-layered communication protocol for load balancing in large scale multi-sink wireless sensor networks.” In “ISADS’09,” pages 223–230 (2009).
- [19] A. Junkie. “Kohonen’s self organizing feature maps.” [Online; accessed Sept-2011] (2011).
- [20] Wikipedia. “Dominating sets — wikipedia, the free encyclopedia.” [Online; accessed 20-April-2012] (2012).
- [21] M. Hassinen, J. Kaasinen, E. Kranakis, V. Polishchuk, J. Suomela, and A. Wiese. “Analysing local algorithms in location-aware quasi-unit-disk graphs.” *Discrete Appl. Math.* **159**(15), 1566–1580. ISSN 0166-218X (2011).
- [22] J. Czyzowicz, S. Dobrev, T. Fevens, H. González-Aguilar, E. Kranakis, J. Opatrny, and J. Urrutia. “Local algorithms for dominating and connected dominating sets of unit disk graphs with location aware nodes.” In “Proceedings of the 8th Latin American conference on Theoretical informatics,” LATIN’08, pages 158–169. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-78772-0, 978-3-540-78772-3 (2008).
- [23] M. Hassinen, J. Kaasinen, E. Kranakis, V. Polishchuk, J. Suomela, and A. Wiese. “Analysing local algorithms in location-aware quasi-unit-disk graphs.” *Discrete Appl. Math.* **159**(15), 1566–1580. ISSN 0166-218X (2011).
- [24] Wikipedia. “Semaphore (programming) — wikipedia, the free encyclopedia.” [Online; accessed 1-May-2012] (2012).
- [25] Wikipedia. “Mutual exclusion — wikipedia, the free encyclopedia.” [Online; accessed 1-May-2012] (2012).
- [26] U. Wilensky. “Netlogo user manual.” [Online; accessed 1-May-2012] (2012).
- [27] Netlogo. “Netlogo user community models.” [Online; accessed 1-May-2012] (2012).