
Cluster Based Synchronous Localized Distributed Algorithm for Load and Latency Balancing



MS Thesis

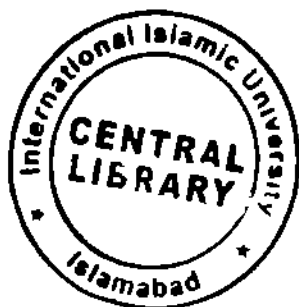
By

Hassan Zeb

704-FBAS/MSCS/S13

Supervisor

Dr Rashid Bin Muhammad



**Department of Computer Science & Software Engineering
International Islamic University, Islamabad**

February 4, 2016

TH-16696 ^{NYA}

Accession No

MS
004.36
HAC

1. Computer algorithms
2. Electronic data processing -
Distributed processing

A Dissertation
Submitted in the Partial Fulfillment of the
Requirements for the Degree of
MS Computer Science

Dedication

This Thesis is dedicated to

My Family and Friends specially to my Father and Mother who support me morally and economically I also dedicated this work to my all teachers specially Dr Rshid bin Muhammad, who guide me in every step of my Thesis to complete it

Acknowledgements

I bow my head before Almighty Allah, who gave me understanding, courage and patience to complete this research. All respects and regards to the Holy Prophet Muhammad (P B U H), who came as a light of knowledge for all seekers. I feel highly obliged in my deepest and sincerest gratitude to my respected teacher and dignified supervisor **Dr. Rashid Bin Muhammad** for his keen interest, benevolent attention, vital and intellectual suggestions throughout the research work. My deepest gratitude and regards to my family for their affection, prayers, financial and moral support. I also pay thanks to Mr. Anwar Ghani for their cooperation.

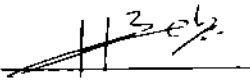


Hassan Zeb

704-FBAS/MSCS/S13

DECLARATION

I hereby declare that this thesis, neither as a whole nor a part of it has been copied out from any source except where otherwise acknowledged in the text. It is further declared that I have prepared this dissertation entirely on the basis of my personal efforts made under the supervision of my supervisor Dr. Rashid Bin Muhammad. No portion of the work, presented in this dissertation, has been submitted in the support of any application for any degree or qualification of this or any other learning institute.

Signature 

Hassan Zeb

MS (Computer Science)

Reg No 704-FBAS/MSCS/S13

Department of Computer Science & Software Engineering

Faculty of Basic and Applied Sciences,

International Islamic University Islamabad, Pakistan

Final Approval

Date - 2-11-2015

Approved by the Department of Computer Science & Software Engineering, Faculty of Basic and Applied Sciences, International Islamic University in partial fulfillment of the requirement for the MS Degree in Computer Science

COMMITTEE

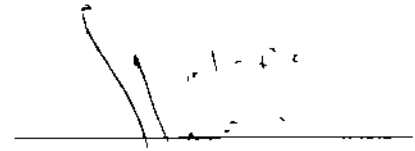
External Examiner

Dr Muazzam Ali Khan
Assistant Professor,SEECS
NUST Islamabad



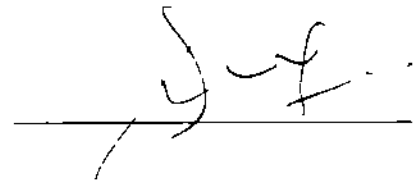
Internal Examiner

Dr Syed Husnain Abbas Naqvi
Chairman, DCS & SE
FBAS.IIU.Islamabad



Supervisor

Dr Rashid bin Muhammad
Assistant Professor,
FBAS.IIU.Islamabad



Abstract

In this work, we have described an algorithm for clustering in ad hoc networks. We have proposed to combine three approaches, BFS algorithm, Lowest ID clustering and minimum outgoing edge. Our goal is to minimize the number of clusters, the number of hops between the nodes and the communication overhead (for broadcasting). Our motivation for this work is to present an efficient sparse network structure for communication. The sparsity is checked through cluster overlaps and node degree. In literature, we have determined that many researchers emphasized on network load (i.e. node degrees) and network latency (the radius). Some of them tried to produce minimum or maximum clusters such as one hope or k-hope clustering. Overlaps and non-overlaps clusters have also been considered by these researchers. Some researcher emphasized on minimum or maximum nodes degrees. We have determined that many algorithms produced clusters with overlaps and if there are no overlaps then the communication overhead (node degree) is not tried to reduce. We have presented an algorithm that is balanced in term of clustering and communication overhead. The cluster creation is based on a condition (i.e. for $k \geq 1$, the number of nodes in the next layer is greater or equal to nodes in the previous tree nodes) and we have proof by lemmas that each cluster consists a tree (which mean that each node has one parent node and hence we have minimized the degree of the node). Our proposed work generate the clusters, which may be one hope or k-hope to gain our purpose. The first node is given which act as the cluster head, the remaining cluster heads are selected with minimum node ID. Our algorithm can be deployed in areas where no infrastructure is available. We have compared our work with previous literature and the results show that our proposed scheme is better in network clusters and network load. A table is also presented for comparison with several algorithms and we have found that our proposed work is good in term of cluster overlaps, node degree, hope counts (between cluster head) and cluster radius.

Contents

List of Figures	iv
List of Algorithms	v
List of Tables	vi
1 Introduction	1
1.1 Distributed System	1
1.1.1 Applications of Distributed Systems	1
1.1.2 Distributed System Model	2
1.1.3 Issues in Distributed Systems	2
1.1.3.1 Communication and Incomplete Knowledge	2
1.1.3.2 Failure, Complexities and Programming Difficulties	3
1.2 Local Information Based Distributed Algorithms	3
1.2.1 LP-Representation	3
1.2.1.1 Clustered Representation	3
1.2.1.2 Skeletal Representation	4
1.2.2 Basic Idea of LP-Representation	4
1.3 Notions and Keywords	4
1.3.1 Thesis Organization	5
2 Preliminaries and Definitions	6
2.1 Preliminaries and Definitions	6
2.1.1 Broadcasting on Network	6
2.1.2 Broadcasting	6
2.1.3 Convergecasting	7
2.1.4 Downcast and Upcast	7
2.1.5 Application of Convergecast and Upcast	7
2.1.6 Time Complexity	8
2.1.7 Space Complexity	8
2.1.8 Message Complexity	8
2.1.9 Communication Complexity	8

2 1 10	Complexity Bounds	9
3	Literature Review	11
3 1	Related Work	11
3 1 1	NCQA Algorithm	11
3 1 2	Cluster Algorithm Based on Parameters	12
3 1 3	K-hope Cluster Algorithm	13
3 1 4	Routing with Guaranteed Delivery (RGD)	14
3 1 5	Cluster Algorithm Based on Node performance	14
3 1 6	Distributed Graph Algorithm (DGA)	15
3 1 7	Optimal Distributed Algorithm for Minimum Weight Spanning Tree	15
3 1 8	Hierarchical Clustering Algorithm	15
3 1 9	Distributed Cluster Algorithm	16
3 1 10	Distributed Sparse Cover Algorithm	17
3 1 11	Adaptive Clustering for Mobile Wireless Network	19
3 1 12	Broadcast	21
3 1 12 1	Distributed Broadcast Algorithm	21
3 1 12 2	Distributed Broadcast Algorithm 2	21
3 1 13	Broadcasting on Preexisting Tree and Non-existing Tree	23
3 1 13 1	Distributed Multi-Message Broadcasting	25
3 1 14	Convergecast	26
3 1 14 1	Distributed Convergecast Algorithm	26
3 1 14 2	Randomize Convergecast Algorithm	27
3 1 15	Downcast	28
3 1 16	Upcast	29
3 1 17	Tree Construction	30
3 1 17 1	BFS	30
3 1 17 1 1	Analysis of BFS	30
3 1 17 2	Distributed DFS	31
3 1 17 3	Distributed Minimum Weight Spanning Tree	31
3 1 18	Summary of Literature Review	32
3 1 19	Problem Statement	33
4	Methodologies	34
4 1	Network Models	34
4 1 1	Communication Model	34
4 1 2	Computational Model	34
4 1 3	Message Passing Model	36
4 1 4	Synchronous Model	36
4 1 5	Asynchronous Model	37

4 1 6	Propagation Model	37
4 1 7	Free Space Propagation Model	37
4 1 8	Outdoor Propagation Model	38
4 1 9	Indore Propagation Model	39
4 1 9 1	Path loss Model (in same Euclidean Plane)	39
4 1 9 2	Partition Loss (Between Euclidean Planes)	40
4 1 10	Log Distance Path Loss Model	40
4 1 11	Unit Disk Graph Model	40
4 1 12	Fresnel Zone Geometry	41
4 1 13	Complexities of Algorithm	41
4 1 14	Other Models	41
5	Proposed Solution	42
5 1	Problem Formulation	42
5 2	Introduction	42
5 2 1	General Overview	43
5 2 1 1	BFS	43
5 2 1 2	DFS	43
5 2 1 3	Broadcast, Convergecast, and Upcast	44
5 2 1 4	Minimum outgoing Edge	44
5 2 1 5	Structure of the Solution	44
5 2 2	Distributed Cluster Algorithm	44
5 2 2 1	Example	49
6	Results and Discussion	50
6 1	Correctness and Complexity	50
6 1 1	Correctness	50
6 1 2	Complexity Analysis	51
6 1 3	Comparison With Other Algorithms	52
7	Conclusion and Future Work	55

List of Figures

3 1	Cover Algorithm Example	19
3 2	Network	20
3 3	Clusters	20
3 4	Broadcasting on a Graph	25
5 1	Cluster Creation	49
6 1	Cluster Comparison	53
6 2	Degree Comparison	54
6 3	Comparison	54

List of Algorithms

3 1 1	NCQA Algorithm	12
3 1 2	Cluster Algorithm on Parameters Based	13
3 1 3	K-Hope Clustering Algorithm	14
3 1 4	Cluster Algorithm Based on Node performance	15
3 1 5	Cluster Algorithm	17
3 1 6	Distributed Sparse Cover Algorithm	18
3 1 7	Cover Algorithm	18
3 1 8	Adaptive Clustering Algorithm	20
3 1 9	Broadcasting	22
3 1 10	Broadcasting on without existing tree	24
3 1 11	Randomize Broadcast Algorithm	26
3 1 12	Distributed Convergecast Algorithm	27
3 1 13	Randomize Convergecast Algorithm	28
3 1 14	Downcast	28
3 1 15	Ranked Items	29
3 1 16	Ordered Items	29
3 1 17	Unordered Items	29
3 1 18	Tree Construction	30
3 1 19	Distributed Depth First Search Algorithm	31
4 1 1	Message Passing Algorithm	36
5 2 1	Cluster Creation Algorithm	47

List of Tables

4 1	Average Signal Loss Measurements Reported by Various Researchers for Radio Paths Obstructed by Common Building Material	39
4 2	Average Signal Loss Measurements Reported by Various Researchers for Radio Paths Obstructed by Common Building Material	40
6 1	Comparison	52

Chapter 1

Introduction

1.1 Distributed System

In a centralized system, the data can be obtained from one location. In distributed system, there may be more than one processors active at the same time and processors can obtain data from each processor. The communication between these processors is done through messages. These systems may be different in size, activities and power. In other words, in a distributed environment, each system has different architecture, size, power, work and each node/system has their own processor and protocols. The protocols are run and make a computation on each processor. Distributed computing uses the message passing model, in which each processor makes an interaction with the other processors through messages. In this thesis our emphasis will be on point to point distributed message passing. In other words, we have only information about our own system and our neighbors', a loosely coupled system uses the messages to our neighbors to communicate with each other. In a tightly coupled system (Parallel), shared memory is used for communication. The communication phenomena in distributed systems become difficult as compared with centralized systems. A networked computer contains a hardware or software component at different locations, communicating with each other through message passing and is called a distributed system. When one system fails in the distributed system the other systems will never get affected by this failure. This failure will be individually [1].

Definition 1.1.1 (Distributed Systems). - *collection of independent computers appear as a single computer [2]*

1.1.1 Applications of Distributed Systems

Distributed systems can be used in many applications such as the web, online games, health, education and finance etc. Google has transferred their whole system in a distributed environment. The search engine produces one of the most complex distributed environments in computer his-

tory This environment consists of a physical infrastructure of networked computers spread all over the world The Google system consists of a distributed file sharing system to support very large files The collection of stock market data, analyzing the political situation and share trading is very necessary for stock markets, which can be provided by a distributed system

1.1.2 Distributed System Model

Distributed system models can be defined in two categories, one is memory sharing models and the other is message passing model In memory sharing model, the processors obtain the data from one location while, in the distributed system, the processor can obtain their data from each other through message passing In this thesis, our model is based on message passing The communication structure can be defined as, given an undirected graph The edges of the graph will be represented the communication channel and the vertices of the graph represent the nodes

Definition 1.1.2 (Graph). - A graph can be defined as, $G = (V, E)$ where V is the set of vertices and E is the set of edges[1]

The vertices are sometimes called nodes, system or processor The edges are called communication channels and will be bidirectional Directed communication will be done only between neighbors The communication phenomena are as follow, the vertices have ports equal to the total number of degree of vertex and each edge is connected to exactly one port Edge (u, v) is connected to both vertices with $((u, i), (v, j))$ pair, where $1 \leq i \leq deg(v)$ and $1 \leq j \leq deg(u)$ Each vertex has input and output buffer for storing in / out messages and processors read from theirs

1.1.3 Issues in Distributed Systems

There are many issues in the distributed system such as communication (cost, speed), incomplete knowledge, failures, timing, algorithmic and programming difficulties Details of these problems are as follows,

1.1.3.1 Communication and Incomplete Knowledge

Sending of information to other nodes do not become free You will be paid for it This is necessary to limit us to speed at which we will propagate the information and also limitations required to the amount of information to be transmitted The given resources should be used wisely In a centralized model, each system is aware of what happened before and after execution, while, in distributed system, each node has only information about their neighbors

1.1.3.2 Failure, Complexities and Programming Difficulties

In a centralized model, if a hardware or software failure occurs then find the failure source and fix it while, in distributed model, the situation is complex. Mostly link failure or lost of packets occurs which is difficult to find. For synchronous and asynchronous models, separate programming for each event such as computation, message passing etc will have to be done. For example, for a synchronous model, programming for local computation and sending a message, while in asynchronous, the programming of the algorithm must be event driven i.e. at event x do y. Time and message complexity of distributed algorithms for distributed system is one of the major issues.

1.2 Local Information Based Distributed Algorithms

The traditional network protocols maintained a table of information in which each node maintains information about the whole network. These protocols become very expensive in term of memory and hence, difficult to maintain in networks. Some researchers realize that global information is not essential for the network. Many tasks can be done by keeping only information about the neighbors. For this purpose we will use Locality preserving network representation approach.

1.2.1 LP-Representation

The main purpose of this approach is a structure which faithfully represents the topology of the network. The purpose of this structure is to maintain only the information about a few edges and a subgraph instead of the whole graph which can reduce the computational and storage cost. Namely, there are two types of LP-Representation, clustered and skeletal representation.

1.2.1.1 Clustered Representation

In this type of LP-Representation, we divided the whole graph into connected subgraphs and make clusters of each subgraph. In clustering approach, the focus is on two parameters, namely, depth or radius of the cluster and overlapping. A distributed algorithm will be efficient if both parameters are controlled. Hierarchical clustering is used by many researchers.

Definition 1.2.1 (Cluster). - *induced sub graph such that all nodes are connected (within a region or) having same properties [3]*

1.2.1.2 Skeletal Representation

This type of LP-Representation consists of sparse spanning sub graphs such as spanning tree

1.2.2 Basic Idea of LP-Representation

In global algorithm environment, the algorithm cannot find the information related to a specific area of the network. Such algorithms visit all the vertices of the graph. The local information based algorithms can restrict themselves to a specific area. For this task cluster technique is used, in which we can maintain information to a specific region. All clusters contain connected nodes and nodes can participate in any activities such as keeping information of the neighbors. A node can participate within cluster activities. As discussed in 1.2.1.1, the two parameters will be controlled to provide an efficient cluster algorithm. A small radius of a cluster provides a low time complexity and a low overlapping algorithm guaranteed low memory complexity. Low message complexity is based on control of both parameters. The other LP-Representation, i.e. skeletal representation, can minimize the edges of the graph. Only one algorithm is used without changing. The parameter used in skeletal representation is the number of edges which refers to the cost of the algorithm.

1.3 Notions and Keywords

Many mathematical notions and keywords will be used in this thesis. The source for notions and keywords are taken from [4, 5, 6]. A Set can be described as, a group of objects. Objects may be a number, symbols or even another set. The object in the set is called element or numbers. For example, a set of the English alphabet. The set will be written in curly brackets. Each set will be presented by a name. A sequence of objects refers to a list of these objects in order. The list will be written within parenthesis. The sequence can be divided into two categories, finite and infinite. A finite sequence is referred to a tuple. There may be k-tuple of sequence element. A two-tuple sequence element called a pair. A relationship between object is called a function. A function takes an object as input and produces an object as output. The input values called domain and the output value called range notation for the function is, $f: X \rightarrow Y$. A Graph can be represented as, $G = (V, E)$ where E is set of edges and V is a the set of nodes. The number of edges incident to a node is called the degree of the node and represented by $deg(V)$. A graph may be directed or in directed. An edge uv from node u to v in a directed graph will be represented as \vec{uv} . While in undirected graph, the edge between nodes u and v will be represented as \overline{uv} . The cluster will be represented as capital \mathcal{A} , \mathcal{B} , \mathcal{C} etc and the covers as A , B , C .

1.3.1 Thesis Organization

Rest of the thesis is organized as, the chapter 2 is based on some definitions and preliminaries. Chapter 3 is based on literature review and chapter 4 is based on methodologies in which we have discussed models that are used in our problem solution. Chapter 5 is based on our proposed solution and chapter 6 is based on results and discussion. Chapter 7 is based on conclusion and future work direction.

Chapter 2

Preliminaries and Definitions

2.1 Preliminaries and Definitions

In this section, we will define some notions, concepts and ideas that will be useful to understand the technical details of the thesis. Our definitions are based on [7, 8, 9, 10, 11, 12]

2.1.1 Broadcasting on Network

In distributed network system, a broadcasting operation is a fundamental approach. Many algorithms have been proposed for broadcasting (discussed in section 3.1). Formally, we can define the broadcast problem as follows,

Definition 2.1.1 (Broadcasting). - *Broadcast operation starts from root r_o , which contains a message Msg in its output for transmission. The message Msg is transmitted to all leaf nodes [6]*

Several algorithms have been proposed for broadcasting such as flooding, broadcasting on existing tree and broadcasting on without existing tree. All these algorithms have been discussed formally in section 3.1.

2.1.2 Broadcasting

Given a spanning tree T rooted at r_o , transmitting a message Msg starts from the root node r_o and transmits the message to child nodes. The algorithm works on a preexisting tree and each processor has the information about the tree. If the tree is not existing, then it can be constructed first through many algorithms such as MST. A broadcast algorithm can also construct a tree. Given a graph $G = (V, E)$, to run a synchronous broadcast algorithm on this graph, will produce a tree rooted at r_o . The tree that is produced by this algorithm will be a BFS tree rooted at r_o (see algorithm 3.1.10). While in an asynchronous system it is not necessary that a broadcast algorithm will produce a BFS tree.

2.1.3 Convergecasting

When the broadcast operation completed then how the root node will assume the halting of the algorithm. For this purpose, the convergecast algorithm is using. The basic idea of convergecast algorithm is that, start from the leaf node and send a message to parent nodes. The message may be a piece of information or echo. Through convergecast, we can also make function computations, namely addition, minimum values and maximum values etc. If a node has some input values and wants to make computation then this function computation is called Semi Group Computation. Formally,

Definition 2.1.2 (Semi Group Computation). - A semi group function δ has the following properties,

- 1 The function δ will be well defined for given input
- 2 The function δ holds the associative and commutative property
- 3 The function δ may represent shortly according to the input

A number of convergecast algorithms defined in section 3.1

2.1.4 Downcast and Upcast

Downcast and upcast are not same as Broadcast and Convergecast. This technique is used when there exist different items to be treated individually and can not be combined with root and child nodes. Our focuses will be on message complexity. Details are in section 3.1.15 and 3.1.16

2.1.5 Application of Convergecast and Upcast

The role of convergecast and upcast is not limited. Here we will discuss some of the applications in which upcast and convergecast used

Smallest Element

In this method, a small value will be collect from each node and will be sent to the root node

Gathering and Propagation of Information

In this method, the information will be collected and propagated to the nodes. The basic idea is that, collect the information in the root node and then broadcast it to the child

Distribution of Tokens

In this method n tokens will be distributed to n nodes in the tree. The token size will be $O(\log n)$ bits

2.1.6 Time Complexity

Definition 2.1.3. *The time complexity of the sequential algorithm can be defined as,*

[Time Complexity of Sequential Algorithm] - The number of steps taken by an algorithm from start to end [6, 1]

As discussed in section 4.1.2, in distributed systems, an algorithm is a collection of sub algorithms (protocols) running on different processors. The time that is taken by all sub algorithms is the total time of the algorithm. Formally,

Definition 2.1.4 (Time Complexity of Distributed Algorithms). *- Given synchronous algorithm A, the total number of pulses generated by A during execution from initial processor to the last processor is called time complexity of the distributed algorithm [6, 1]*

2.1.7 Space Complexity

In sequential algorithms, the total space covered by an algorithm is called Space Complexity. In distributed algorithms, the space complexity can be defined as,

Definition 2.1.5 (Space Complexity of Distributed Network). *- The total memory bits covered by algorithm A in the network in the worst case is called space complexity [6, 1]*

2.1.8 Message Complexity

Let T be a tree rooted at r_0 and we wish to send a message Msg on it. Exactly one message will be sent on each edge. The total number of messages sent by an algorithm on each link is called message complexity. If there are n nodes then $n - 1$ messages are sent. In other words,

Definition 2.1.6 (Message Complexity). *- The message complexity of distributed algorithm A is the total number of message bits transmitted during execution of the algorithm A [6]*

In this thesis, the basic length of message will be $O(\log n)$

2.1.9 Communication Complexity

During message passing, delays occur on edges in a given time due to highly traffic load and queued message. We are given a weighted graph $G = (V, E, dist)$ where V is the set of nodes and E is the set of edges and $dist$ is a positive weight associated with each link. A weight function $dist: E \rightarrow Z^+$ is associated with each edge is used to find the expected time that a message will take to send to the destination [1]

Definition 2.1.7 (Communication Complexity). - *Transmitting a message over an edge e is the weight of that edges refers to communication cost. The total cost of the message that is sent during the execution of algorithm A is called communication complexity of algorithm A .*

[13, 6, 1]

2.1.10 Complexity Bounds

A bound on an algorithm refers to the worst case. Usually, the researcher used the upper and lower bounds on the algorithms. These bounds apply to each input of the algorithm. The upper bound refers to a global bound. This bound hold for every input and execution of an algorithm. For example if we say that, the time complexity of the algorithm A is $O(n \log n)$, this mean that there exist a constant $c > 0$ such that $time(A) \leq cn \log(n)$ and when we say that the lower bound of algorithm A is $\Omega(n)$ it means that, there exist a constant $c > 0$ such that $(A) \leq cn$ [6, 1]

Definition 2.1.8 (Unit Disk Graph). - *Unit disk graph can be described as the edges between all nodes having distanced at most 1 [10]*

Definition 2.1.9 (Euclidean Graph). - *a weighted graph where the weights of edge represent Euclidean distances between the adjacent nodes in the plan [7]*

Definition 2.1.10 (Planar Graph). - *a graph that can be drawn in the plan without edges crossing (or crossed edges number are 0) and the intersection is done only on its vertices of G [11]*

Definition 2.1.11 (Delaunay Triangulation). - *A triangulation of S points is a Delaunay triangulation, denoted by $Del(S)$. If the circumcircle of each of its triangles does not contain any other nodes of S in its interior. A triangle called the Delaunay triangle if its circumcircle is empty of nodes of S [8]*

Definition 2.1.12 (Vertex Induced Sub graph). - *A sub graph $H(V', E')$ is called induced sub graph of $G(V, E)$ if $v' \in V$ and $e' \in E$ and each v' has the edges whose ends are v' [9]*

Definition 2.1.13 (Edge Induced Sub graph). *is a graph whose v' set is the end of e' [9]*

Definition 2.1.14 (Gabriel Graph). - *Let disk (u, v) be a disk with a diameter (u, v) . Then, Gabriel graph $GG(S)$ is a graph in which edge (u, v) is present if and only if disk (u, v) contains no other points of S . $GG(S)$ is a planar graph [1]*

Definition 2.1.15 (Gabriel Edges). - *let uv be an edge then uv is called Gabriel edge if $|uv| < 1$ and the disk that has uv as diameter, does not contain any node from S [8]*

Definition 2.1.16 (K-Localize Delaunay Triangulation) - *A triangle ∇uvw is k -localized Delaunay if the interior of disk (u, v, w) does not contain any node of S and all edges of the triangle ∇uvw have a length less than 1 [8]*

Definition 2.1.17 (Localized Delaunay Triangle). - *a k -localized Delaunay triangle for some integer $k \geq 1$ is called Localized Delaunay Triangle [8]*

Definition 2.1.18 (Minimum Spanning Tree). - *Given a connected, undirected graph $G(V, E)$, a spanning tree of that graph is a sub graph that is a tree and connects all the vertices of*

the graph. A Minimum Spanning Tree (MST) or minimum weight-spanning tree is a tree that connects all the vertices of $G (E, V)$ with less weight [12]

Definition 2.1.19 (Computational Rounds). - in the synchronous system, total no of rounds is considered as time measuring until termination. These total numbers of rounds refer to time complexity for the synchronous system. For the asynchronous system, rounds can be determined by propagating waves across the network [14]

Definition 2.1.20 (Local Running Time). - determining global running time of a network is difficult. Local running time refers to the computation of an algorithm between different classes of computers and we are required the total running time of each class of computers [14]

Definition 2.1.21 (Close-in-Distance). - the distance that is near with transmitter antenna [15]

Definition 2.1.22 (θ - Notation). - can be defined as, for large size N , the function $f(n)$ be positive [12]. Mathematically, we can write this as, $0 \leq c_1g(n) \leq f(n) \leq c_2g(n)$ for all $n > n_0$.

Definition 2.1.23 (O - Notation). - also called big oh, this notation is used for upper bound [12]. Mathematically, we can write as, $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.

Definition 2.1.24 (Ω - Notation). - can be used for lower bound [12]. Mathematically notation is, $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.

Definition 2.1.25 (Broadcast). the broadcast process is starting from the source node r_0 , the message is kept in the input buffer and then transmit to its child nodes [6]

Definition 2.1.26 (Breadth First Spanning Tree). - a tree T is BFS if the length between the root node and other node v is minimum [6]

Definition 2.1.27 (Cover). - a collection of clusters $S = S_1, S_2, \dots, S_n$ of the graph $G = (V, E)$ is called a cover that consist all vertices of the graph [6]

Chapter 3

Literature Review

3.1 Related Work

In this chapter, we will describe the previous literature that is related to our work. Specifically, our work is based on the [18, 34, 17, 14, 13, 15, 8, 20, 5, 16, 197, 31]

3.1.1 NCQA Algorithm

The authors [16] defined the model of the paper as, given an undirected graph $G = (V, E)$ where V referred to the number of nodes and E to the number of edges. The cluster was defined as the subset of connected nodes. The neighbor of the CH was the nodes that had a connection with it. The author produced the concept of strong nodes, weak nodes and border node. If node v had more than Three neighbors i.e. $deg(v) \geq 3$ then this node was said to be the strong node. If a node had equal to two neighbors then it was weak and a node with one neighbor node called border node. For cluster head selection, the priority given to strong node. Another concept was the division of range in zones, such as strong zone, intermediate zone and risked zone. The first two zones were called the trusted zone while the third zone had the neighbors which had no trusted neighbors. The algorithm was based on two parts, cluster head selection and formation of cluster head member's set. Cluster head selection was based on the weight of the node, here weight mean

$$w_i = D_v W_1 + ST_R W_2 + L_B W_3 + B_L W_4 \quad (3.1)$$

where D_v was the distance factor, ST_R was the stability factor, L_B was the load balancing factor and B_L was the remaining battery time

If the weight was greater then select it as a CH. In the second part, start the construction of the cluster set. In which the author finds the degree of the node. The stability of the cluster means that the node position was not changed during execution. Load balancing factor was used to minimize the number of nodes. The author had calculated the node dissemination degree that reflects the relative deviation of the number of neighbors. We have determined that cluster overlaps are high in this paper and node degree not changed. The algorithm was as follow:

Algorithm 3.1.1: NCQA Algorithm

input : A Graph $G = (V, E)$ neighborhood and distance

output: Set of clusters

```

1 for  $v \in G$  do
2   Find the neighbor of  $v$ 
3    $cal(E)$ 
4   Calculate stability factor
5    $deg(v)$ 
6   Find battery time
7   Calculate  $w_v$ 
8 end
9  $CW \leftarrow u_v$ 
10 Sort  $CW$  in increasing order
11 while  $CW \neq \emptyset$  do
12    $v_i \leftarrow CW, CH \leftarrow v_i$ 
13   Delete  $v_i$  and neighbor of  $v_i$  from  $CW$ 
14 end

```

3.1.2 Cluster Algorithm Based on Parameters

In this paper [17] the author presented an algorithm for clustering which was based on node degree, remaining power level, stationary level and remaining time battery. The assumption of this paper was that, the network topology was time event, when a message was broadcast then each neighbor node received the message from a fixed radio transmission range and each node had fixed ID and know their fitness parameters. The author told that there was seven messages and the most important was *ch_info* message. In *ch_info* message each node downcast the address, fitness, address of CH and node state. Each node maintained the information table which store the information related to the parameter and the internal table to stores the neighbor information. Table updating was based on the message event. When a message was received the table becomes updated. In this paper, the latency is more (as the clusters contain only the neighbor nodes) and also the degree of the nodes is high. Cluster overlaps also occur. The general idea of the algorithm was as follow:

Algorithm 3.1.2: Cluster Algorithm on Parameters Based

input : A Graph $G = (V, E)$ neighborhood and distance
output: Set of clusters

```

1 foreach  $v \in G$  do
2   |  $Bcast(ch\_info) \rightarrow neig(v)$ 
3   |  $cal(F)$ 
4 end
5 if  $cal(F) > neig(F)$  then
6   |  $Post(ch\_nominated) \rightarrow neig(v)$ 
7   | and wait for  $ch\_nominated$  from  $neig(v)$ 
8 end
9 if  $Get(ch\_nominated) = \emptyset$  then
10  |  $CH = v$ 
11  |  $Post(ch\_selected) \rightarrow neig(v)$ 
12 else
13  | Again compare
14 end

```

The formula for fitness was as follow

$$F = D_1 W_1 + P_R W_2 - S - F W_3 + B_L W_4 \quad (3.2)$$

where D_1 was the number of degree of the node, P_R was the received power level, $S - F$ was the stationary factor and B_L was the remaining battery time

3.1.3 K-hope Cluster Algorithm

Chen et al [18] described a cluster based algorithm. In which he created clusters by the lowest ID of the nodes. The basic idea was that, select any node for starting the algorithm and transmit the message Msg to k-hope neighbors. The author takes into account the degree of the nodes. This mean that if a node has high degrees among k-hope neighbors, then select this node as the cluster head otherwise if in case of a tie, chose the minimum ID node. This algorithm reduced the number of clusters (for k) but not good when k is one. As a resultant the broadcasting overhead (communication complexity) become low but the node degree or load was still a problem. Formally, the algorithm can be defined as follow

Algorithm 3.1.3: K-Hope Clustering Algorithm

input : A Graph $G = (V, E)$ where V was set of nodes and E was set of links
output: Set of Clusters

```

1 Post(my_id)
2 upcast(my_id, deg(v))
3 if my_id < rec_ids and my_deg(v) ≥ rec_deg(v) then
4   | mc_id = my_id
5   | broadcast(my_id, mc_id)
6 else
7   | if my_deg(v) > rec_deg(v) then
8     | | mc_id = my_id
9     | | broadcast(my_id, mc_id)
10  | end
11 end

```

3.1.4 Routing with Guaranteed Delivery (RGD)

Bose et al [19] were the first who presented average-case efficient routing algorithms with guaranteed delivery in ad hoc wireless Networks. In the first algorithm, extract a connected planar sub graph in distributed manner. The main idea was, intersect the Unit Graph with Gabriel Graph and the author showed by a lemma that the resultant sub graph was planar. In the second algorithms, they presented the routing algorithms, one was Face-1 and the other was the (extension of Face-1) Face-2 which was better in performance than Face - 1. The routing was done by famous Left-Hand rule.

3.1.5 Cluster Algorithm Based on Node performance

The authors [20] were described an algorithms that were based on the performance of the node. The performance could be evaluated through Residual energy, free memory, processor speed, disk space and node density. The formula was

$$\text{Nodepref} = \text{Residual energy} \times b + \text{free memory} \times b + \text{processor speed} \times b + \text{disk space} \times b + \text{node density} \times b$$

Where b was a weight and can be found by the classification rank order centroid method. The main algorithm same as OLSR protocol but the selection of the cluster head was different. In OLSR protocol, the nodes were divided into three states, state 0 (not decided), state 1 (cluster head) and state 2 (member) furthermore, each node calculate their own performance and send it to the neighbors. The algorithm was as follow,

Algorithm 3.1.4: Cluster Algorithm Based on Node performance

```

1 state= ∅
2 cal(nodepref)
3 if nodepref < nodeperfi then
4   | mystate= CH
5 else
6   | mystate= member
7 end

```

3.1.6 Distributed Graph Algorithm (DGA)

In this [21] paper, the problem can be divided in two phase Phase I extract the connected planar sub graph from the given graph by the geometric routing algorithm in ad hoc wireless network. The problem can be formulated in the geometric graph as follows. Let N be a set of nodes deployed in a certain region R , with $|N| = n$. The problem was to build a planar graph $G = (N, E)$ on N such that each node was connected to its closest neighbor. Formally the edge $(u, v) \in E$ if and only if $\delta(u, v) \leq 1$. Where $\delta(u, v)$ was the distance between node u and its closest neighbor v . The author used to extract a localized Delaunay triangle from a graph G and remove other edges. Phase II In this phase, the author used face routing algorithm of Kranakis et al [22] and Bose et al [19] for routing on the graph computed by the algorithm in phase I.

3.1.7 Optimal Distributed Algorithm for Minimum Weight Spanning Tree

Awerbuch [23] presented distributed algorithms for MST. Two stages was used in this algorithm. Counting Stage in this stage spanning tree can be found, another purpose of this stage was to find the total number of nodes and a leader can be selected in the network. MST Stage the total number of nodes can be sent to MST stage that funded in the first stage and using this information find minimum weight-spanning tree. The message and time complexity for both stages were $O(V + V \log(V))$ and $O(V)$ respectively. The main purpose was that, The small tree was not waiting for big trees.

3.1.8 Hierarchical Clustering Algorithm

Banerjee et al [24] described a structure of hierarchical clustering. There was two phases of the algorithm, first phase based on BFS algorithm to create a spanning tree rooted at r_0 , and the second phase was based on the cluster creation procedure. When a BFS tree formed then piggyback a message to the root to start the cluster creation process. The root is the lowest id

node in the graph. The author showed that two cluster heads could be neighbors in the subgraph. The cluster creation procedure was as follows:

Each node discovered their subtree size and adjacency information of the children in the tree. The piggyback message contains information related to subtree size and adjacency information. The piggyback message started from the leaf nodes to the root nodes. When a node realizes that its subtree becomes larger than k , then it stops and makes the cluster. If the number of nodes becomes smaller than k in the subtree, then the cluster formed by the algorithm could be one.

Mamalis et al. [25] prove that this algorithm produces clusters with overlaps. It is noted that the node degree is still not changed. We also note that in some clusters, the hop or distance between two nodes becomes large.

3.1.9 Distributed Cluster Algorithm

In this section, we will show some algorithms that were used for sparsity. We will consider some parameters such as neighborhood relation between clusters and the outgoing edges. In the next section, we present a clustering algorithm which was based on [26].

The aim of the cluster algorithm was to divide the graph into subgraphs. A partition $S = \{S_1, S_2, \dots, S_i\}$ was a collection of disjoint clusters, i.e., $S \cap S' = \emptyset$ where $S, S' \in S$. Due to disjoint clusters, there was no overlap between clusters. Informally, the algorithm starts from a given source node and constructs clusters layer by layer. In the layering procedure, each node would be the part of some cluster. The parent nodes in algorithm 3.1.10 would be the leader nodes which formed the clusters. The child nodes would first find their parent and then the parent node would create the cluster.

Algorithm 3.1.5: Cluster Algorithm

```

input : A set of Clusters
output: A set of Covers
1  $S \leftarrow \emptyset$  and  $p=0$ 
2 while  $V$  is not empty do
3   select  $v \in V$ 
4   if  $v$  was parent or root then
5      $S \leftarrow v$ 
6     while  $p \leq \text{deg}(v)$  do
7        $S \leftarrow \text{neigh}(v)$ 
8     end
9      $S = S \cup S$ 
10     $V = V - S$ 
11  end
12 end
13 output  $S$ 

```

3.1.10 Distributed Sparse Cover Algorithm

In distributed computing sparse neighborhood cover was first introduced in the paper [27]. A sparsity can be achieved by controlling the diameter and the overlapping of the clusters. If we consider the diameter of a cover r in the cover then overlapping of clusters become more and if we consider the whole graph/network as one cover then the diameter become uncontrolled (as up to n where n was the number of nodes). So the author bound the r -neighborhood cluster diameter to $O(r \log n)$ and cluster overlapping to $O(\log n)$. The time and space complexity was $O(n \log n)$. The main idea had been taken from [27]. A cover $S = \{S_1, S_2, \dots, S_i\}$ was a collection of clusters $\mathcal{S} = \{S_1, S_2, \dots, S_i\}$. The main idea of the algorithm was, divide a given graph $G = (V, E)$ in a connected subgraphs, called clusters and combine some clusters to form a cover. This cover would consist all the vertices of the graph. Some problem arises in covers such as when neighbors nodes belongs to separate clusters (which cause information exchange problem). To solve this problem, a 1-2 approach would be used. Two basic parameters were used for a cover for quality evaluation, namely, the size of clusters and Level of interaction. If the size of the cluster was small then it was possible that the communication within cluster become less i.e. message complexity. The radius of the cluster define the size. Interaction level was defined by the degree of the node / cluster. The algorithm was based on the idea of "Coarsening" i.e. repeatedly merge the clusters to form a cover. Here the author discussed two algorithms, namely, cover algorithm and Max cover algorithm. Max cover algorithm make covers from the clusters and the cover algorithms make clusters. Informally, the cover algorithm starts from the root cluster and construct an output cluster Y iteratively. The root cluster Y was

called kernel of the cluster that would be constructed in next iteration a new layer Z would be constructed around Y which hold those clusters which were not the part of merged clusters. The merging process was based on layering. The cluster that was built after the root would be considered as a kernel and so on. The merging process would stop after sparsity condition i.e. $|Z| \leq n^{1/k} |Y|$

Formally, we can define the cover algorithm as follow,

Algorithm 3.1.6: Distributed Sparse Cover Algorithm

input : A graph $G = (V, E)$ Cluster \mathcal{H} and integer $K \leq 1$

output: Cover Clusters S

- 1 $R \leftarrow S$
 - 2 Set S to empty
 - 3 **Repeat**
 - 4 $(CUM, CM) \leftarrow \text{merge}(R)$
 - 5 $S \leftarrow CM$
 - 6 $R = R - CUM$
 - 7 **UNTIL** $R = \emptyset$
-

The algorithm (3.1.6) starting from assigning clusters to \mathcal{H} and the cover \mathcal{H} to empty. In the loop, the merge function was called. This function was used to find clusters that joined the Y clusters and the remaining clusters. After finding the merged clusters, we add these clusters to cover. The process terminated when there was no remaining cluster. Now, let formally define the procedure *merge*.

Algorithm 3.1.7: Cover Algorithm

input : A set of Clusters

output: A set of Remaining and Merged Clusters

- 1 $\mathcal{U} = \mathcal{H}$
 - 2 set CM and CUM to empty
 - 3 **Repeat**
 - 4 Select $\mathcal{S} \in \mathcal{U}$ and set \mathcal{Z} to \mathcal{S}
 - 5 **Repeat**
 - 6 set \mathcal{Y} to \mathcal{S}
 - 7 Set $\bigcup_{Y \in \mathcal{Y}} \mathcal{S}$ to Y
 - 8 $\mathcal{Z} \leftarrow \{s \text{ where } s \in \mathcal{S} \text{ and } S \cap Y \neq \text{empty}\}$
 - 9 **Until** $|Z| \leq n^{1/k} |Y|$
 - 10 $\mathcal{U} \leftarrow \mathcal{U} - \mathcal{Z}$
 - 11 $CUM \leftarrow CUM \cup \mathcal{Y}$
 - 12 $CM \leftarrow CM \cup \mathcal{Y}$
 - 13 **Until** $\mathcal{U} = \emptyset$
 - 14 **Output** (CM, CUM)
-

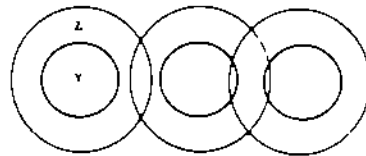


Figure 3.1 Cover Algorithm Example

The algorithm 3.1.7 starting from the unprocessed clusters U equal to \mathcal{X} . Some cluster of \mathcal{U} merged to another cluster Y iteratively starting from root cluster i.e. the root cluster refers to Y and in each iteration, some cluster of U merged with Y . Y was also called the kernel. The clusters from U was intersected with Y and merged in a layered fashion with Y which produce a new cluster Z . For the next layer, the Z cluster would be viewed as a new kernel. The process would continue until a sparsity condition becomes true. The resultant cluster Z then add to CM while Z contain the kernel clusters. \mathcal{Y} contain those clusters which were not the part of the kernel and added to CUM .

Time and Space Complexity

Theorem 3.1.1. *The algorithm 3.1.7 construct a*

1. *Coarsening Cover CM which coarsens CUM and*
2. *There was no common vertex between kernels i.e. $Y \cap Y' = \emptyset$ and*
3. *$CUM \leq |\mathcal{X}|^{1-1/k}$ where*

Proof 1. This was clear from the beginning of the algorithm 3.1.7 that the input was connected clusters so each cluster Y added to CM was clusters.

2. Let suppose for a contradiction that $\exists v \in Y \cap Y'$ then v belongs to Y and Y' by definition. This situation cause a contradiction, because of $\mathcal{Y} \cap \mathcal{Y}' \neq \emptyset$ condition in the algorithm. v must be in Y not in next kernel by eliminating from \mathcal{U} in last steps of the algorithm.

3. As we can see from the termination condition of the internal loop that $|\mathcal{X}'| \leq n^{1-k} |Y|^{1/k}$. $|\mathcal{Y}|^{1/k} = |\mathcal{X}'|^{1-1/k} |CUM| = CUM \geq |\mathcal{X}'|^{1-1/k}$.

□

3.1.11 Adaptive Clustering for Mobile Wireless Network

Lin and Gerla [28] presented an algorithm which was based on lowest ID of nodes. Informally, pick the nodes that have the lowest ID, mark this node as the first cluster and broadcast the cluster ID to neighbors. The neighbor nodes would check their ID with the received cluster ID, if the node ID was greater than the received cluster id then merged in the cluster otherwise make

your own cluster. Each node broadcast the cluster message before halting. The problem of this algorithm is the maximum number of clusters and degrees. Formally, the Distributed Cluster algorithm was as follow,

Algorithm 3.1.8: Adaptive Clustering Algorithm

input : A Graph $G = (V, E)$ where V was set of nodes and E was set of links

output: A set of Clusters

```

1 if  $my\_id$  was smallest in the neighbors then
2   |  $mc\_id = my\_id$ 
3   | broadcast ( $my\_id, mc\_id$ )
4 end
5 on receiving ( $my\_id, mc\_id$ )
6    $rc\_id = mc\_id$ 
7   check with own id
8 if  $mc\_id$  was empty or  $my\_id > rc\_id$  then
9   |  $mc\_id = rc\_id$ 
10  | remove my id from the list
11 else
12  | repeat the first procedure
13 end
    
```

Example of Adaptive Clustering Algorithm

The example of the algorithm 3.1.8 is as follow

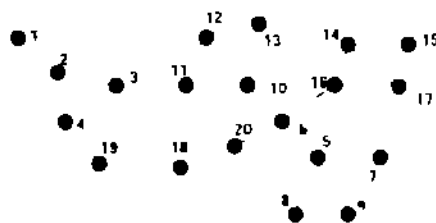


Figure 3.2 Network

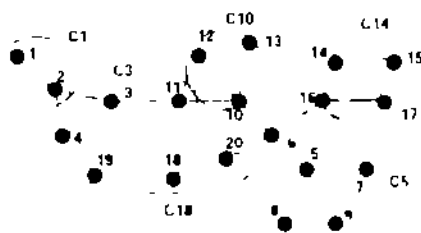


Figure 3.3 Clusters

3.1.12 Broadcast

In flooding, we resend multiple copies of a packet with hope that at least one copy of message would reach the destination but this causes a maximum load on the network. Broadcasting was used for one-way communication from the parent node to all its child nodes. In broadcasting, each node would send a message/packet exactly one time. In this section, we will discuss some algorithms that are helpful for our work.

3.1.12.1 Distributed Broadcast Algorithm

Awerbuch [26] also presented a distributed algorithm. He combined two algorithms one of which have low time complexity but communication complexity was high and the second one has low communication complexity but time complexity was high. Awerbuch try to make such algorithm that was fine in both time as well as in communication complexity. A node was safe if each message was sent and received at that pulse in which it send. Informal description of the first algorithm was as, through acknowledgment, each node detects that it was safe and report this to its neighbors and when the root node determined that it all Child was safe then it generates the next pulse. The communication complexity was $O(|L|) = O(|V|^2)$ and the time complexity was $O(1)$. Description of the second algorithm was as, at initialization phase, a leader s can be chosen and the spanning tree would be constructed rooted at r_0 . First the leader would try to understand that all the nodes were safe in the current pulse and broadcast the message that the leader may generate a new pulse. Now when a node becomes safe then it sends a message to ancestors that he was safe (i.e. convergecast). The message complexity was $O(|V|)$ and the time complexity was $O(|V|)$. Note that the time was proportional to the height of tree which may become $|V|-1$ in the worst case. The proposed algorithm was as follow in the initialization phase, the whole network can be divided into clusters. The tree in one cluster was called intracluster tree. The communication between clusters would be done on one link that was the minimum weight link. A leader would be chosen inside each cluster and this leader would be responsible for operation between intracluster trees.

3.1.12.2 Distributed Broadcast Algorithm 2

Atiya and Welch [5] also presented a distributed broadcast algorithm in which each node receive Msg at most D time. The main assumption was that, the network topology would be fixed and the message exchange between nodes must be constant. The idea was that, when each child node receive the message Msg , it sends to the next neighbor (child) nodes except the sender node. A single bit was used to check where the node saw this message or not, if yes then terminate the algorithm. Formally the broadcast algorithm was as follow.

Algorithm 3.1.9: Broadcasting

input : A Graph $G = (V, E)$ where V was set of nodes and E was set of links

output: A spanning tree rooted at r_0

```

1 if I am the root then
2   | Seen-Message  $\leftarrow$  True
3   | forward message Msg to all Neighbor
4 else
5   | if I am the non-root then
6   |   | Seen-message  $\leftarrow$  false
7   |   | Receive the message Msg
8   |   | if seen-message  $\leftarrow$  false then
9   |   |   | seen-message  $\leftarrow$  true
10  |   |   | send message Msg to all other nodes
11  |   |   end
12  |   end
13 end

```

Correctness

Theorem 3.1.2 (Correctness of 3.1.9). *Each node would receive at most $|L|$ messages after at most D time from v . Where E was the set of edges and D was the diameter of the network.*

Proof Take the first part of the theorem. The root node would send the message to all its connected edges and the child node would send the message to its child except parent node. So this scenario lead us to the point that the message would travel across all the edges of the graph once. If a node is not received the message then it means that this was not the neighbor of any node. Now let proof the next part of the theorem by induction.

- 1 **Basis** If there was no neighbor of the root i.e. $\text{dist}(\text{root}, v) = 0$, Then root would receive the message at time 0. Therefore the first step was true.
- 2 **Inductive step** for inductive step let $d(\text{root}, v) = k$ where $k > 0$. Since $\text{dist} \neq 0$ which means that there exist the neighbor of v i.e. u so,
 - $\text{dist}(\text{root}, v) = k$
 - $\text{dist}(\text{root}, u) = k - 1$ and
 - $\text{dist}(\text{root}, v) = (k - 1) + 1 = k$ Hence the proof

□

The message complexity of 3.1.9 algorithm was $|E|$ and the time complexity was D as discussed in 3.1.2 theorem.

3.1.13 Broadcasting on Preexisting Tree and Non-existing Tree

The broadcast algorithm problem can be divided into two categories, namely broadcasting on existing tree and without existing tree [5] we only discuss broadcast without existing tree. This algorithm was also based on flooding technique. Three variables involved in this algorithm

- *parent* // Broadcasting from root
- *child* // Broadcasting from leafchildren
- *already* // I am already someone's child

Informally Attiya and Welch [5] described the flooding algorithm as follow, each node maintain their own processor and local memory. The computational process was same as discussed in section (4.1.2). Each node contains *inbuf* and *outbuf* for each channel additionally with a *parent*, *children* and *other* variables. The root node would be set to *parent* p_i node and deliver the message *Msg* to the children p_j nodes. If a p_i node received the message *Msg* for the first time from p_j node then it would set the *parent* variable to p_j and the message would be forwarded to next neighbor node. After arriving a message from other node p_k , if p_i node received already a message from p_j and p_k , select the p_j node as *parent* then it would send a message *already* to p_k that I have already selected a parent. Possibly, a node can receive more messages from several nodes, then it would select their *parent* arbitrarily among them. Some variables were used in the algorithm such as *other*, *parent*, *child*, *myparent* and *terminate*. The algorithm was as follow. Initially *other*, *parent* and *child* variables was empty.

Algorithm 3.1.10: Broadcasting on without existing tree

input : A Graph $G = (V, E)$ with a distinguish root node r_v
output: A BFS tree rooted at r_v and each node have their parent

- 1 At the root p_r node
- 2 **if** *outbuf* is empty **then**
- 3 | *halt* \leftarrow true
- 4 **else**
- 5 | Parent = p_r
- 6 | δ Outbuf _{i} \times deliver \rightarrow inbuf _{j}
- 7 **end**
- 8 At child node the message delivered from p_j node
- 9 **if** *parent* \leftarrow \emptyset **then**
- 10 | *parent* = p_j
- 11 | Post(Ack) to p_j as parent
- 12 | Compute(t)
- 13 | δ inbuf \times compute \rightarrow outbuf _{j}
- 14 | deliver(i, j, msg)
- 15 | δ Outbuf _{i} \times deliver \rightarrow inbuf _{$i \rightarrow j$}
- 16 **else**
- 17 | Send *myparent* to p_j
- 18 **end**
- 19 On receiving Ack from child p_i
- 20 *child* \cup p_i
- 21 **if** *children* \cup *other* contain all neighbors except parent node **then**
- 22 | *halt*
- 23 **end**
- 24 On receiving *myparent* from node p_i
- 25 *other* \cup p_i
- 26 **if** *children* \cup *other* contain all neighbors except parent node **then**
- 27 | *halt*
- 28 **end**

Figure 3.4 is an example of the 3.1.10 algorithm. In this example, the red edges show that this is not the edge of the resultant tree. While the black line shows that the edges are the part of the tree.

Lemma 3.1.3. *The algorithm 3.1.10 Produce a BFS tree rooted at P_r .*

The author showed by induction of t that the resultant tree of algorithm 3.1.10 was a BFS tree.

Basis

For $t = 1$

Since all variables are empty and the message is in transit mode from p_r and there is no node in

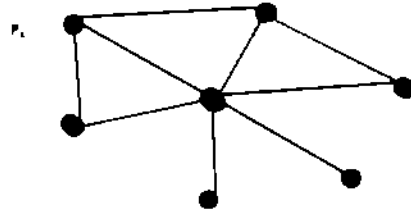


Figure 3.4 Broadcasting on a Graph

this position

Inductive step

Let this is true for $t - 1 = 1$. During round $t - 1$, Let f be a node that receive the message Msg from the parent node f at distance $t - 2$. The node f received the message after distance $t - 1$. The node that received the message at distance $t - 2$ not changed its parent position neither delivered a message and every node at distance $t - 1$ changed his parent as well as deliver the message. Thus, we can say that each node connects only those node which is at distance t i.e. neighbors. This lemma shows that the message complexity is $O(M \cdot g)$ and time complexity is $O(D)$.

3.1.13.1 Distributed Multi-Message Broadcasting

Yu et al. [29] proposed a randomized algorithm for broadcasting, in which he used a threshold $(\mu \log n)$ probability for time slots to send a message to the next node. Informally, the algorithm works on dominating set and start from any leader randomly. The leader propagates the message and the non-leader hear from the parent. If a leader received the message for the first time from another leader then delete the message from the queue after sending. Formally we can describe the algorithm as follow.

Algorithm 3.1.11: Randomize Broadcast Algorithm

```

input : A Dominating Set
output: Propagation of information
1 if Msg  $\neq$  empty then
2   | for  $\mu \log n$  time-slots do
3   |   | deqeu  $\leftarrow$  Msg
4   |   | Post  $\rightarrow$  next_leader
5   |   end
6 else
7   | Listen from other nodes
8 end
9 Upon Get(Msg)
10 if Get(Msg) then
11   | Buffer the message Msg and
12   | Post(Msg)
13 end

```

3.1.14 Convergecast

The convergecast algorithm was used for two-way communication from child nodes to the parent node. The main purpose was to collect information throughout the network. The process of convergecast was done in opposite manner of the broadcast.

3.1.14.1 Distributed Convergecast Algorithm

Peleg [6] defined the converge cost process in the following words. The algorithm starts from the leaf nodes and send the inputs to its ancestors nodes recursively. Formal description of the convergecast algorithm was as follow,

Algorithm 3.1.12: Distributed Convergecast Algorithm

```

input : A tree
output: Collection of information
1 if  $P_{id} = Leaf$  then
2   | Send input to parent node
3 end
4 on receiving message Ack and input  $\exists$  from leaf node Buffer = ( $\exists$ , Ack)
5 if  $buffer \neq \emptyset$  then
6   |  $v \leftarrow f(\text{buffer}, \text{input})$ 
7   | if I am the root then
8     | Return v
9   | else
10  | Send v to parents
11  | end
12 end

```

Theorem 3.1.4 (Correctness and Complexity of 3.1.12). *the message complexity of the 3.1.12 algorithm is $O(n-1)$ and the time complexity is D*

Proof The proof is same as theorem 3.1.2 but the only difference is that, the messages send by leaf nodes are only $n-1$ Where n is the number of nodes of the network □

3.1.14.2 Randomize Convergecast Algorithm

Kesselman and Kowalski [30] presented a randomize distributed convergecast algorithm. The basic idea of the algorithm was that, initially all nodes was active and then starting the algorithm to collect data. The collection of data was based on rounds at one time step. A transmission range of an active node can be defined as, the closest distance between nodes. So set the transmission range and transmit the data to the neighbor at one time step and then the node become inactive. At last, when one active node remains then the algorithm become halt. Formally the distributed convergecast algorithm can be define as,

Algorithm 3.1.13: Randomize Convergecast Algorithm**input** : A Graph $G = (V, E)$ where V is set of nodes and E is set of links**output**: Collection of data from all nodes

```

1 if my state is active then
2   for round  $t$ , start do
3      $Tran(R) = \min \text{dist}(u, v)$ 
4     if  $Post(Msg)$  without any collision then
5        $My\_mode = active$ 
6     else
7       end
8     merge the received data with self
9   end
10 end

```

3.1.15 Downcast

In downcast, each root has distinct items $I = \eta_1, \eta_2, \dots, \eta_n$, which was destined in a specific node in the tree. The item may be a number or message [6]. The algorithm for downcast was as follow,

Algorithm 3.1.14: Downcast**input** : A tree with root node r_0 **output**: Downcasting of distinct messages

```

1 if status is root then
2   Send the message to the subtree rooted at  $w$  one by one on edge  $(r_0, w)$  in order
3 else
4   if status = intermediate then
5     receive at most one message at each step
6     if status = destination then
7       don
8     else
9       Send to next node  $u$ 
10    end
11  end
12 end

```

Lemma 3.1.5. Algorithm 3.1.14 take time $O(Msg + dept(T))$ on Msg distinct messages on the tree T

Proof In each step the root r_0 propagate at least one message. So it will take time Msg . When

the message are never delayed, it proof the lemma as message start from r_o and leave at time t . The message reach to its destination after $Msg + dept(T)$ □

3.1.16 Upcast

In upcast, since each root has distinct items $I = \eta_1, \eta_2, \dots, \eta_n$, which was destined in one or more node of the tree. Each item of the node would be sent to parent node individually. The items may be ranked, ordered and unordered. the goal was to collect information on the root node.

Ranked items - all items are taken from an ordered set and given rank to each item. The upcasting was done in following manner [6],

Algorithm 3.1.15: Ranked Items

input : A tree rooted a r_o with distinct data items

output: Upcasting ranked items

- 1 At first round
 - 2 **if** The i th (i, I_i) item is stored in local node **then**
 - 3 | forward to parent node
 - 4 **end**
-

Ordered Items In this case the items has been taken from ordered set but not ranked. This type of items used for comparison. The upcasting was done in the following manner,

Algorithm 3.1.16: Ordered Items

input : A tree rooted a r_o with distinct data items

output: Upcasting the smallest item

- 1 On each round
 - 2 **if** I am the smallest item **then**
 - 3 | forward to parent node
 - 4 **end**
-

Unordered items the items was taken from the unordered set. No comparison can make but only for ID i.e. only send IDs of the node to the parent [6]

Algorithm 3.1.17: Unordered Items

input : A tree rooted a r_o with distinct data items

output: Forward local stored item to parent

- 1 On each round
 - 2 Forward the item to parent if not upcast
-

77-16696

3.1.17 Tree Construction

A variety of algorithms introduced by various researcher for construction of trees such as BFS, DFS and Distributed Shortest Path

3.1.17.1 BFS

To construct a BFS tree, There are two types of distributed BFS tree construction, one is Dijkstra and the other is Bellmon-ford Let we examine the Dijkstra The distributed Dijkstra algorithm construct the BFS tree T_p in layer by Layer manner from the root node to neighbor nodes in each pulse At each layer, the parent node adds the neighbor nodes that is not in the tree yet Informally we can describe the distributed Dijkstra algorithm as follow [6]

Algorithm 3.1.18: Tree Construction

input : A Graph $G = (V, E)$ with a distinguish root node r_o
output: A layered BFS tree rooted at r_o

- 1 At root r_o node
- 2 $pulse = 0$
- 3 $bcost(Msg) \rightarrow neig(v)$
- 4 On child node
- 5 $Get(Msg) = true$
- 6 **if** $parent(w) = \emptyset$ **then**
 - 7 | $parent(w) \leftarrow v$
 - 8 | $v \leftarrow child(w)$
 - 9 | $collect(Ack) \in neig(v)$
 - 10 | $upcast(Ack)$ to r_o
 - 11 | start next pulse
- 12 **else**
- 13 | $Post(Ack)$
- 14 **end**

3.1.17.1.1 Analysis of BFS

Time Complexity Let analyze the time and message complexity of the above algorithm The broadcast and convergecast process would take $2D$ time The downcast and upcast (find new neighbors) would take 2 additional time units, The total time is $time(\text{phase } p) = 2p+2$

for $1 \leq p \leq \text{diam}(G)$, then

$$\sum_p \text{time}(\text{phase } p) = \sum_p 2p + 2 = O(\text{diam}^2(G))$$

Message Complexity There are p iteration and the total messages on each iteration is $O(n)$. Let $p \geq 0$ and V_p, E_p be the set of vertices and edges repetitively of the sub graph $G(V_p)$ in a layer p . Let E_p be the internal edges of V_p , and let $E_{p, p+1}$ be the edges connecting V_p to V_{p+1} . At phase p , exploration messages are sent only over E_p and $V_{p, p+1}$, and the edges of E_p are traversed twice, giving $\text{message}(\text{phase } p) = O(n) + O(|E_p|) + O(|E_{p, p+1}|)$

for $1 \leq p \leq \text{diam}(G)$, then

$$\sum_p \text{message}(\text{phase } p) = \sum_p O(n) + O(|E_p|) + O(|E_{p, p+1}|) = O(n * \text{diam}(G) + |E|) [6]$$

3.1.17.2 Distributed DFS

This algorithm was commonly used for exploration of a graph [31, 6]. Basic idea of BFS was, start from root r_0 and send a message to the neighbors if the child was visited by another node then return to parent otherwise visit it and processed further. If we visit a node that have no any parent then done. In distributed version, a token was used to traverse the graph in depth first manner. Informally, let examine the DDFS,

Algorithm 3.1.19: Distributed Depth First Search Algorithm

input : A Graph $G = (V, E)$ with a distinguish root node r_0 .
output: A DFS tree rooted at r_0 and each node have their parent

- 1 Start the token from a parent node r_0 ,
 - 2 When the token arrived at child node
 - 3 **if** *status* = *unvisited* **then**
 - 4 | Explore them ,
 - 5 **else**
 - 6 | Go back to parent r_0 ,
 - 7 **end**
 - 8 *parent(v)* = NULL, then we terminate since $v = r_0$.
-

3.1.17.3 Distributed Minimum Weight Spanning Tree

Gallager and Bui et al [32] had presented algorithms that built a distributed MST. The main idea was that every node start with a single fragment and each node find the shortest distance edge. Each fragment has a level. The fragment that has one node was at level 0. The level of fragments used for the combination of fragments. If a fragment was wished to connect with another fragment then we have the following cases. If $L = 1$ where F was the fragment of L and $L' = 2$ where F' was the fragment of L' then F would be merged in F' and the next level

will be $L' + 1$. If F and F' have the same shortest distance edge and $L = L'$ then the fragments from a new fragment F'' at level L'' and the edges was called the core edge. The nodes have three states, the sleeping, find and found states. The sleeping state was the initial state of the node, the find was the state when a fragment search its minimum outgoing edge and the found state was the state when the minimum outgoing edge during convergecast, each node upcast its minimum weight of edges through an intermediate node and when the convergecast terminate at the root node, the root selects the tuples with minimum weight and this was the MWOGE. When a node was awoken, it find its minimum edge and marks this edge as branch. Then it send a connect message to another fragment through branch edge and then it goes to found state. The algorithm take a fragment and connect with other fragment through the core edge. The weight of the edge was the identity of the fragment and act as a root of the fragment tree. Nodes that was adjacent to the core sent an initiate message to borders. On arrival of initiate message the node, find the minimum outgoing edge. This node send a test message to all the edges. If the edges were same then it reject, if the level was high then accept and if the level was low then wait. Every node send the report. If no smallest edges found then the algorithm was complete. Message complexity was $\theta(E + \sqrt{\log V})$ and time complexity was $(\sqrt{\log V})$. The problem was, small trees wait for big trees and this was solved by [23].

3.1.18 Summary of Literature Review

In this section, we are going to present the summary of the literature review. We have discussed in chapter one that we will present a distributed algorithm which will be based on Local Preserving Representation (LP Representation see section 1.2.1.1). The cluster representation of LP Representation is our main point of concentrate (section 1.2.1.1). In chapter four we have discussed some models that will be used in our Thesis. The computation model 4.1.2 presents the communication phenomena between nodes. As we discussed in chapter one that the communication between distributed systems done through messages, the message passing model (section 4.1.3) will be used as communication model in our thesis. Section 4.1.11 and 4.1.14 are also involved in our problem. In chapter Two, we have discussed some definitions, from which we have got some ideas to understand our problem. Mamali et al [25] presented a clustering algorithm which solves the problem of intracluster as well as the intercluster communication. They have used the BFS algorithm to find the clusters and when the tree constructed then start the upcast procedure to build the clusters. Also communication between nodes become less but in some cases, we have noted that the intracluster communication becomes large. In other words, the communication between two nodes (or cluster radius) become large as we can see in figure 6.3a. The algorithm of Aissa et al [16] is useful in term of a number of clusters of the network. It maximizes the number of clusters of the graph. But there are some problems with this algorithm. The radius of the network become large and also the intracluster as well as the intercluster communication has become costly. The cluster overlaps cannot be stopped in

this algorithm. Chen et al. [18] presented a clustering algorithm in which the communication between cluster minimized but still there is the problem of cluster overlaps and degree of the nodes not reduced. The author used two techniques to find the cluster head, one is lowest ID, and the other is maximum degree. This algorithm is good when k is greater than three otherwise our presented algorithm is the good solution. In papers [26, 27] presented clustering algorithms based on k -hope layers. The authors of these papers considered the total number of nodes and compare it with the previous nodes to find a sparse structured network but the problem is high communication complexity. Papers [29, 5, 26] are based on broadcast and convergecasting which are involved in our thesis. DBFS and DDFS algorithms [6, 31] also have involved in our thesis.

3.1.19 Problem Statement

We have determined in the literature review that many papers are based on load balancing and other papers emphasize on latency balancing but if we minimize the load then the latency become maximum and if we minimize the latency then the load become maximum. We need to balance both load and latency.

In other words, we can define our problem as

Given a weighted network graph $N = (S, L, dist)$ on the 2-dimensional Euclidean space, where S is the set of n sensors, L is the set of communication links among sensors such that $s_1, s_2 \in S$, and a non-negative weight function $dist: E \rightarrow Z^+$. Our thesis (problem) is to build a distributed sparse structure for efficient communication in network N in a localized manner on the distributed computational model (synchronous and asynchronous) such that the resultant network N is sparse subnetwork and balanced in terms of network load and network latency.

Using the assumptions, we proposed an algorithm that will make balanced the above mentioned two criteria. Our algorithm will use the clustering technique in the synchronous distributed environment to balance the load and latency for the wireless networks.

Chapter 4

Methodologies

4.1 Network Models

In this section we will discuss the model that will be used throughout the thesis. The main papers and books for this section are [13, 12, 5, 2, 6, 4, 1]

4.1.1 Communication Model

Point to point communication network will be represented as an undirected graph $G = (V, E)$ where $V = \{v_1, v_2, \dots, v_n\}$ as a set of nodes/processors and $E = \{e_1, e_2, \dots, e_n\}$ as a set of edges representing communication bidirectional channels connecting the nodes. Sometime a non-negative weight function $dist: E \rightarrow Z^+$ will be associated with the edges $e = (u, v)$ connecting u, v nodes. Each node/processor will be identified by a unique ID $P = \{p_1, p_2, \dots, p_n\}$ which is one to one mapping $\tau \in ID: V \rightarrow P$. The communication phenomena is as follow the vertices have ports equal to the total degree of vertices and each edge is connected to exactly one port. Edge (u, v) is connected to both vertices with $((u, i), (v, j))$ pair, where $1 \leq i \leq deg(v)$ and $1 \leq j \leq deg(u)$. Each vertex has input and output buffer for storing in / out messages and processors read from theirs.

4.1.2 Computational Model

The computation is based on algorithm followed by protocols $\tau \in A = \{A_1, A_2, A_3, \dots, A_n\}$ and each protocol A_i is running on each v_i processor. Each protocol A_i consist set of states, a set of alphabets and a function such that $A_i = (\Sigma, Q_i, \delta, Q_0)$

Where

$$\Sigma = (\text{inbuf}, \text{outbuf}) \in \text{Msg}$$

$$Q_i = (Q \wedge Q)$$

$$\delta = \delta Q_i \times \Sigma \rightarrow Q_i$$

$Q_0 =$ initial state

There is state set (\overline{Q}_i) for edges and each edge is from the state \overline{q}_i . \overline{q}_i consist two components for directions, $v \rightarrow u$ and $u \rightarrow v$. Let Msg denote messages that are to be used by the processor in the execution of the algorithm. Then $v \rightarrow u$ is an element of Msg . $(\overline{q}_i \rightarrow u = Msg)$ mark that the message is going from u to v and $v \rightarrow u = \emptyset$ denote that the channel is empty. At the starting, all the processors are at the initial stat and the channel is empty. The distributive execution of algorithms is based on events. The events may be a local computation or message event. Moreover, two events, local computation (internal operation i.e. changing of states), send (message send) or deliver (the message buffered in the destination input) are used in events. Distributed system consist sequence of configuration which shows the current state of the processor and the link. Formally we can define the configuration as follow, configuration consist a tuple $C = \{q_1, q_2, q_3, \dots, q_n, \overline{q}_1, \overline{q}_2, \overline{q}_3, \dots, \overline{q}_m\}$. q_i is the processor local state and \overline{q}_j is the edge e_j state. Computation of algorithm is a sequence $Sq = \{C_0, \sigma_1, C_1, \sigma_2, C_2, \sigma_3, \dots\}$ of configuration. Here C_1 refers to the configuration with initial configuration C_0 and σ_k refers to the event. At this level, the computation is occurring as [6]

$$r_0 = C_0$$

$$\delta = (C_i, \sigma_i) = C_{i+1}$$

Attiya and Welch [5] discussed the computational model as, each node contained a distinguish channel to the parent node and a set of channels to the neighbors node. As discussed above, each algorithm has a sub algorithm in each processor for local computation such as sending and receiving messages and also each processor consist a set of states Q_i . Each state of processor contains special components, *inbuf* and *outbuf* for every edge labeled from 1 to $deg(v)$. The transition function takes a value from the state of p_i and produces a value as output. This function also can produce at most one message for every link between 1 and $deg(v)$ and this message to be sent to the other node. The configuration is modeled as a vector $C = \{q_1, q_2, \dots, q_n\}$ where q_i is a state of p_i (as the change in state, change of tape and change of head location in Turing machine). The occurrence is modeled as an event. Possibly there are two types of events, computation event $compute(i)$ for transition function and delivery event $deliver(i, j)$ *msg* for message delivery.

In this section, we will study the computational model that are directly or indirectly include in our work.

4.1.3 Message Passing Model

In message passing model, one processor sends a message to other processors to communicate with each other over a communication channel. These communication channels are bidirectional (symmetric) which lead us to topology. This topology can lead us to an undirected graph. Therefore, the processors referred to nodes and there is an edge between two nodes *iff* there is a communication channel between processors (nodes). For example, on the Internet, we consider routers as nodes, they are communicating with each other through IP address and IP packets are used for exchanging of data. Simply we can say that, message passing model has a set of processes having only local memory, communicating with each other by sending/receiving messages and for sending of data both send/received must cooperate with each other [14]. Lynch [13] described message passing model as a collection of process and each process has states, which is related to a set Q_i of states. *inbuf* and *outbuf* component are used for message available for sending and the message is ready for outgoing. Each process has a configuration (change in state). A configuration occurs when an event exist i.e when a message delivered or a new message is added to *outbuf*. A message sending algorithm in message passing model is as follow,

Algorithm 4.1.1: Message Passing Algorithm

```

1 On Client Computer
2 do
3  $R_1 \leftarrow Msg$ 
4  $Post(Msg) \rightarrow Server$ 
5 On Server Computer
6 if  $Get(Msg)$  then
7    $R_2 \leftarrow Msg$ 
8    $Post(Ack) \rightarrow client$ 
9 end

```

4.1.4 Synchronous Model

Muhammad [14] put it, in the synchronous model clock was used for sending and receiving a message. An internal clock was used for sending the message between processors. This time was same for all processors. Lynch [13] told that each processor/node send a message at a time t and on the other side each processor could be received this message in $t+1$ time. $t+2$ times would be used for next processor to send the message and so on. The whole process runs in lockstep.

4.1.5 Asynchronous Model

In this model, there was no fixed time for the nodes to send and receive messages. We can say that if there is no upper bound that how much time a message will take to arrive on the receiver side then this is called asynchronous. A clear example is email service. There are no assumptions for clock (same clock time for all processor). Only conditions or events can be used for the algorithm to perform. E.g. FIFO, FILO etc. [14]. Minimum restriction by the arrival of a new message or local computation event occurred. Some of the assumptions are, each process will take some finite computation steps and each message will be delivered finally. Every distributed computing use the client server model. In which clients send a request to the server and the server response to the client.

4.1.6 Propagation Model

Signals can be transmitted by transmitter antenna and these signals can be received by receiver antenna. This model is useful for path loss. Disturbances in receiving a signal is referred to path loss. Propagation model can be divided into two parts. One is Statistical Model which is based on analysis of curves by measured data. e.g. Okumura Model, Hata Model, and Lee Model. Other is a physical model, which is based on electromagnetic waves, which is concerned with indoor propagation model also called site-specific-Model [15].

4.1.7 Free Space Propagation Model

In free space propagation model, observation of received signal strength is determined when there are no disturbances (i.e. reflection, diffraction and scatter). This free space equation is used for free space propagation model.

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{4\pi^2 d^2 L} \quad (4.1)$$

Where P_t is the transmitted power, $P_r(d)$ is the received power, G_t is the transmitter gain power, G_r is the receiver antenna received power, d is the T-R-Separation distance in meter, L is the system fault (antenna losses during communication) and λ is wave length in meters. Gain power is based on aperture of the antenna [15].

$$P_r(d) = \frac{4\pi A_e}{\lambda^2} \quad (4.2)$$

Where A_e is the physical size of the antenna

The equation for path loss in free space path loss with antenna gain power become

$$PL \text{ (dB)} = 10 \log \frac{P_t}{P_r} = -10 \log \frac{G_t G_r \lambda^2}{(4\pi)^2 d^2} \quad (4.3)$$

Moreover, when the antenna gain power is excluded then the equation becomes

$$PL \text{ (dB)} = 10 \log \frac{P_t}{P_r} = -10 \log \frac{\lambda^4}{(4\pi)^2 d^2} \quad (4.4)$$

Fris free space equation is suitable for to predict the values of receiving power $P_r(d)$ for values of d where d is the far field of the transmitting antenna. The far field, which is also called Fraunhofer region, is the farthest distance from the antenna. Equation for Fraunhofer region is

$$d_f = \frac{2D^2}{\lambda} \quad (4.5)$$

Where D is the largest dimension of antenna [15]. Fris free space equation does not hold for $d = 0$. Large scale propagation model use close-in-distance. So the newly received power equation is

$$P_r(d) = P_r(d_0) \left(\frac{d_0}{d}\right)^2 \quad d \geq d_0 \geq d_f \quad (4.6)$$

Where d_0 is the close in distance [15]

4.1.8 Outdoor Propagation Model

During finding the path loss, one thing will be taking into account that the irregular lands, trees, and buildings will affect this. These lands may start from unbalanced land to high mountains. Many researchers proposed different propagation models to predict path loss. The main concentrate of these researchers is a particular local area, i.e., a sector or region such as Lon-

gley rice model [33], which is also called irregular terrain model. This model is suitable for point-to-point communication in irregular land between 40 MHz to 100 GHz range. Through geometrical shapes, the path loss can be determined. Okumura Model [34] is suitable for the frequency range between 150 MHz to 1920 GHz. This model is simple and good for path loss predictions [15]. Hata model [35] is best in 150 MHz to 1500 MHz. Prajesh and Singh [36] told that Hata and Okumura are better in suburban areas and the Longley rice model in rural areas.

4.1.9 Indoor Propagation Model

This model is different from traditional mobile radio Channel in two aspects, small distance and small range of TR-Separation distance. Propagation mechanism is same as Outdoor i.e. reflection, diffraction and scattered. To find the path loss, the condition change from outdoor model as weak signal strength due to building material, open/closed doors and building type etc. Some models that use indoor propagation model are following.

4.1.9.1 Path loss Model (in same Euclidean Plane)

In the construction of building different type of material used. In which some walls are made by concrete and some made through woods that divide the walls into small pieces of spaces. The partition of walls refers to hard partition (just like concrete walls, which become the permanent structure of the wall) and soft partition (that can be move or not become the permanent structure of the building, e.g. woods etc) [15].

Table 4.1 Average Signal Loss Measurements Reported by Various Researchers for Radio Paths Obstructed by Common Building Material

Material Type	Loss (dB)	Frequency	Reference
All metals	26	815 MHz	[37]
Aluminum siding	20-4	815 MHz	[37]
Foil insulation	4-9	815 MHz	[37]
Concrete block wall	13	1300 MHz	[38]
Loss from one floor	20-30	1300 MHz	[38]
Loss from one floor and one wall	40-50	1300 MHz	[38]
Fade observed when transmitter turned a right angle corner in a corridor	10-15	1300 MHz	[38]
Tight textile inventory	3-5	1300 MHz	[38]
Chain-like fenced in area 20 ft high containing tools, inventory and people	5-12	1300 MHz	[38]
Metal blanket 12 sqft	4-7	1300 MHz	[38]
Metallic hoppers which hold scrap metal for recycling 10 sqft	3-6	1300 MHz	[38]
Small metal pole 6 diameter	3	1300 MHz	[38]
Metal pulley system used to hoist metal inventory a 4 sqft	6	1300 MHz	[38]
General machinery 10-20 sqft	5-10	1300 MHz	[38]
Light machinery E 10 sqft	1-4	1300 MHz	[38]

4.1.9.2 Partition Loss (Between Euclidean Planes)

The structure of the buildings floors may affect the path loss between floors due to different materials, construction type and external structure even number of windows and doors

Table 4.2 Average Signal Loss Measurements Reported by Various Researchers for Radio Paths Obstructed by Common Building Material

Building	915 MHz FAF (dB)	1 (dB)	Number of Locations	1900 MHz	1 (dB)	Number of Locations
Walnut Creek						
One Floor	33.6	3.2	32	25	31.3	110
Two Floors	44.0	4.8	39	38.5	4.0	29
SF PacBell						
One Floor	13.2	9.2	16	26.2	10.5	21
Two Floors	18.1	6.0	10	33.4	9.9	21
Three Floors	24.0	5.6	10	35.2	5.9	20
Four Floors	27.0	6.1	10	38.4	3.4	20
Five Floors	27.1	6.3	10	46.4	3.9	17
San Ramon						
One Floor	29.1	5.8	93	35.4	6.4	4
Two Floors	36.6	6.0	81	35.6	5.9	41
Three Floors	39.6	6.0	70	35.2	3.9	27

4.1.10 Log Distance Path Loss Model

As the distance increased, the average received signal power from the transmitter antenna decrease logarithmically in both indoor and outdoor models

$$PL(\text{dB}) = PL(d_0) + 10n \log\left(\frac{d}{d_0}\right) + X_n \quad (4.7)$$

Where n is the path loss exponent, depend on different environment such as free space (in which $n = 2$) and obstruction present (n may be large) $PL(d_0)$ is the path loss w.r.t d_0 and X_n is a variable with normal distribution in dB

4.1.11 Unit Disk Graph Model

The model that is used in this paper is the Unit Disk Graph model i.e, let (V, E) be a set of the pair where V is the set of nodes and E is the communication link. Moreover wireless ad hoc network used a mutual transmission range for all nodes. If node u, v are in this range then an edge $e \in E$ exist between u and v . The Mutual Transmission Range lead us to the definition of Unit Disk Graph, let $G = (V, E)$ be a graph then this graph is called Unit Disk Graph if an edge $(u, v) \in E$ present in G where the distance between $|u - v| \leq 1$ [21]

4.1.12 Fresnel Zone Geometry

Fresnel Zone Geometry can be defined as the radius between transmitter and receiver

$$r_n = \sqrt{\frac{n\lambda d_1 d_2}{d_1 + d_2}} \quad (4.8)$$

Where r_n is the radius of the Fresnel Zone, n is an integer, λ is wavelength d_1 is the distance from transmitter antenna and d_2 is the distance from received antenna

4.1.13 Complexities of Algorithm

In sequential algorithms, the performance of the algorithm can be evaluated through time and space complexity. While in distributed environment the performance evaluation criteria become change

4.1.14 Other Models

Different models can be studied to check the behavior of distributed systems. Some models deal with, fault tolerance, dynamic changes in network and studying nature of time etc. In this thesis, we will keep self to the models that contain the following characteristics

1. Ignore fault handling issues. Thus, we can assume that the network is fault free and all the processors are reliable. Moreover will work on the static network and dynamic changes will not be considered
2. Each node will contain a unique ID of $O(\log n)$ bits
3. Local computation on each processor will be free

Also in this thesis, we will ignore some parameters i.e. weights (unweighted graph). The focus will be on these three models, namely LOCALITY MODEL, CONGEST MODEL AND ASYNCH MODEL assumed from above discussion. The focus of LOCALITY MODEL is on the localize nature of execution. One assumption is that communication between nodes will be synchronous and computation will be done in the same round. The focus of CONGEST MODEL is on the size of the message and the total message size will be $O(\log n)$. This model works on both synchronous and asynchronous model. The focus of the last ASYNCH MODEL is on asynchronous communication. In which both sized and unsized messages will be considered [13, 6]

Chapter 5

Proposed Solution

5.1 Problem Formulation

In this chapter, we will discuss the assumptions on which our thesis problem stands and formally define the problem. This thesis studies the communication problem in networks in which nodes know only their label. The nodes of the network required to building an efficient communication structure so that any operation for algorithm takes time less than the time primitive steps. This results in the efficient implementation of the algorithm on the communication structure or network. From the graph theoretic viewpoint, we consider nodes of the network as the set of vertices and treat edges as communication links. The distance among communicating nodes are treated as standard weight function. In general, our goal is to construct a communication network in which one can control the load and the latency, which are one of the two most important criteria for efficient communication in the network. Formally, our assumptions are as follows.

Assumptions

- Nodes know nothing but their own unique id and the id of their neighbors
- All nodes $s_i \in N$ are symmetric, that is,
$$\forall s_1, s_2 \in S, \text{dist}(s_1, s_2) = \text{dist}(s_2, s_1)$$
- Given a root node r_0

5.2 Introduction

In this section, we are going to discuss a distributed algorithm for clustering. As discussed in section 1.2, LP-Representation is used to maintain a small amount of information just like information of few edges and structure of the graph which can control the computational and space cost. The cluster is a type of LP-Representation, in which a specific amount of nodes

grouped in a small connected subgraph. In clusters, we control the radius and overlaps between clusters in such a manner that the resultant structure is balanced in term of load and latency.

5.2.1 General Overview

We are considering a weighted undirected network $N = (S, L, dist)$ where S is a set of sensors, L is the bidirectional communication links between neighbors and $dist$ is the weight function. The links between neighbors can be established through broadcast and convergecast. An extreme solution of the problem is that each node has one edge but this cause minimum load and more latency. The proposed solution is based on two phases.

Phase I Divide the whole graph into clusters (subnetwork)

Phase II And ensure that each cluster contains a tree

Shortest Outgoing Edge (SOE) is used to find the small edges that is related to each cluster. The Distributed Prior Wide Search algorithm is suitable for to extract the local tree. The problem of BFS can be defined as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges of directed rooted at a node r_0 (any) spanning tree. We assume that the network is strongly connected and a root node r_0 is exist which is not same with others. The out put of the algorithm will be a directed rooted tree at source node r_0 .

5.2.1.1 BFS

The BFS algorithm is used in this solution from the start and algorithm will be called again and again as the message *Msg* triggered. The basic assumptions of our BFS algorithm is as follow.

Assumptions Each node knows its *id* and neighbors *id* and nothing else such as load, latency or the length of the diameter of the network.

5.2.1.2 DFS

The algorithm (3.1.19) is used to find the remaining nodes in the graph. The remaining node refers to those nodes which are not the part of any cluster. For this purpose, a *RETREAT* message is sent to the previous cluster from which it created, through *parent_node*. Informally the algorithm operates as follow: go back to the root node through *parent_node* (the node from which it received the *test* message) and traverse the TRELs which is built through the cluster create procedure.

5.2.1.3 Broadcast, Convergecast, and Upcast

In these algorithms, we can send and receive a message particularly in upcast we can find a minimum id (as describe in section 3.1.14) The tree that is built in clusters is achieved through a broadcast algorithm, described in section 3.1.12 and 3.1.14

5.2.1.4 Minimum outgoing Edge

When the algorithm reached to a layer that is empty then we go back to the root node by the node which is selected as the parent nodes The parent nodes are those nodes which is sending the message $LAYER(p,t)$ to the next layer nodes

5.2.1.5 Structure of the Solution

Summarized form of the algorithm is as follow,

Given a graph $G = (V, E, dist)$ and an integer $k \geq 1$

1 Cluster creation procedure

In cluster creation procedure, the clusters creation starts through BFS tree algorithm When the sparsity condition of our problem become false then create the cluster otherwise increase the layer to next nodes

2 Next leader selection procedure

When the first cluster created then which node will be the next cluster head? For this purpose, the next_leader procedure started to select that node which has the smallest node in the neighbors

3 Minimum outgoing edge selection procedure

When there is no node in the graph without a cluster, then find the minimum edge in the intracluster edges

5.2.2 Distributed Cluster Algorithm

In this section, we present a clustering algorithm that uses a condition for sparse structure The basic idea has been taken from [32, 6, 31, 26, 18, 27, 39] The algorithm creates clusters in iterations At each iteration, a BFS tree is created and all the neighbor nodes add to the cluster with a certain sparsity condition informally, the algorithm starts from the root node and trigger a message *Leader node* to itself and then *Msg* message to itself After *Msg* message the BFS algorithm starts in layer by layer fashion At each layer all the node are added to the cluster until a sparsity condition is satisfied i.e the nodes in the new layer is less than previous all layer nodes (see line 23 of the algorithm on page 47) This is our basic condition which decides the

tree height (means number of nodes in the tree) If the condition becomes true then expand the tree else if the condition becomes false then reject the new layer nodes and make the cluster from the previous nodes. When the algorithm determined that there is no nodes in the new layer then our algorithm become halt. The tree that is built in the cluster is a BFS tree with t_n . A number of messages used in this procedure such as *Msg* message, *LAYER (P-1, l)*, *Ack(bit)*, *COUNT* message and *REJECT* message. As the root node generated the pulse P , BFS tree created at the root with $P-1$ layer. Each pulse maintains a number which leads to the decision of adding or rejecting the layer to the cluster. The joining of $P+1$ layer to the cluster is based on the condition, compare the child nodes in the $P+1$ layer with the nodes in the previous all nodes of BFS tree, if the ratio is $\geq n^{1-k}$ then increase the layer to $P+2$ otherwise reject the new layer and create the current cluster. When the leader l broadcast the *Msg* message on the existing tree then the next pulse execution started. When the message reached to the last layer $P-1$ nodes it send a *LAYER (P-1, l)* message to its children that my layer is $P-1$ governed by l . After arriving the *LAYER (P-1, l)* message, the child node J (not yet joined any cluster) join the cluster l at layer P and chose the node from which message *LAYER (P-1, l)* received as a *parent_node* send *ack* (0 in case of parent or 1 otherwise) to l . The total number of the new nodes in the layer can be found through convergecast process by inserting the *COUNT (t)* message. When the leader knows the number of the new nodes then the process become halt and if the number of new node is high than the present number of nodes the next pulse is started and the nodes of the last layer joined the cluster. If the number of nodes not greater than the existing cluster nodes then broadcast a *REJECT* message that the nodes in this layer are rejected. Each cluster has a unique ID. After the message *REJECT* another procedure starts which is called the Next-Leader procedure (see line 3 of the algorithm on page 48). This procedure chose the next leader of the cluster. There is two cases after reject a layer one is when the layer nodes is empty and when the layer is not empty from nodes. If the layer is not empty then a node in this layer is chosen as a leader and if the layer is empty from nodes then backtrack procedure applied to the cluster by DFS algorithm. For the next center of activity the leader l of the cluster called a subroutine search-leader which determines the leader of the next cluster. For this purpose a test message is sent on the intracluster tree by the cluster leader and the last layer that join the cluster broadcast the *test* to the rejected layer. When this message reached to the rejected layer node t , it search their neighbors in remaining graph. A convergecast process is started on the leaf of the node t to find the minimum identity node and set this node as a candidate. A new-leader message is broadcast in the intracluster tree to inform the candidate node that you are the next cluster leader. upon receiving of the new-leader message, the candidate node mark the node as parent node from which the message received and then create their own cluster. For the second case i.e. if the rejected layer is empty, the last created cluster send a *RECREATE* message to the parent node and try to find the remaining node of the graph. *RETREATE* message send to parent node until the parent node become empty i.e. The root cluster, which has no any parent node when the remaining nodes become end then the procedure Cluster-Leader become halt. Link Election procedure is called for the selection of a preferred link for communication between

two nodes (see line 24 on page 48). Informally, a weight function $dist: E \rightarrow Z^+$ is attached with each link. The link between the cluster is chosen during the procedure of backtracking of Cluster-Leader procedure and at this level the nodes attached to some clusters. A message ME is broadcast by the cluster leader on the intracluster tree. Each node in the tree convergencast a list to the leader node which holds the minimum edge (ME) information. When an internal node receives the list it compares their own list and drops the redundant edges. The leader node collects the information and then broadcasts on the tree. Now let us define the algorithm informally.

Algorithm 5.2.1: Cluster Creation Algorithm**input** : A Graph $G = (V, E)$ with a distinguish root node r_0 and a parameter $k \leq i$ **output**: Set of distinct clusters

```

1 if  $i = r_0$  then
2   | trigger (Leader_node)  $\rightarrow r_0$ 
3   | set parent_node  $\leftarrow \emptyset$ 
4   | send message Msg to itself to create the cluster
5 else
6   | /* node  $i$  received Leader node message from node  $j$ */
7   | trigger (Leader_node)  $\rightarrow i$ 
8   | parent_node  $\leftarrow j$ 
9   | leader  $\leftarrow i$ 
10  | layer ( $i$ )  $\leftarrow 0$ 
11  | pulse  $\leftarrow 0$ 
12  | trigger(Msg)  $\rightarrow i$ 
13 end
14 upon receiving message Msg
15 /* start the BFS algorithm*/
16 /*the BFS tree constructed on the root  $r_0$ , at layer 0 and send the the next message Msg to
   next node*/
17 Broadcast(Msg)  $\rightarrow$  child
18 if child =  $\emptyset$  then
19   | trigger(child_nodes) = 0 to parent
20 else
21   | trigger(child_nodes)  $\rightarrow i$ 
22 end
23 if  $k$  (child_nodes)  $\geq$  tree_node then
24   | confirm new layer and send message Msg in next layer
25   | /* improve the intracluster tree*/
26 else
27   | /*reject this new layer (send rej message to leader) and create the cluster*/
28   |  $\mathcal{S} = \text{neigh}(r_0)$ 
29   |  $S = \mathcal{S}$ 
30   | /*remove the nodes that joined the cluster*/
31   |  $S = S - \mathcal{S}$ 
32   | /* complete the intracluster tree (cluster created)*/
33 end

```

```

1 upon receiving message rej
2 if the layer is not empty then
3   | Broadcast(test) →  $T_k$ 
4 else
5   | trigger(RETREAT) →  $Leader_i$ 
6 end
7 upon receiving message test from node k
8 invoke search-leader procedure
9 Broadcast(test) →  $T_k$ 
10 candidate = min{k ||  $k \in \text{neigh}(i)$ }
11 /*search the neighbor node and find the minimum ID node through convergecast process*/
12 upcast(candidate) →  $Leader_i$ 
13 /*upon receiving candidate message in the leader node of the cluster*/
14 broadcast(candidate) →  $T_k$ 
15 if the node i is candidate node
16 if candidate = i then
17   | /* the candidate node is the next leader of the cluster*/
18   | trigger (Leader_node) → i
19   | /*send New_Leader message to itself*/
20   | parent_node ← k
21   | /* k is the node from which i received the message candidate*/
22 else
23   | /* at this level all the nodes join some clusters*/
24   | convergecast the ME message to leader of the cluster
25   | /* start the preferred link procedure*/
26 end
27 upon receiving message RETREAT
28 start the backtracking procedure
29 /*DFS algorithm*/
30 find the remaining nodes of the graph through test message by the leader of the cluster
31 if such node is find then
32 Repeat the test message process
33 upon receiving the message ME on the parent node i
34 send ME message to  $\text{neig}(i)$ 
35 /* find the minimum edge and create a list*/
36 send the list to parent_node

```

```

1 upon receiving list to intermediate node
2 merge list to itself
3 remove the duplicate entry
4 and send to the parent_node
5 if parent_node =  $\emptyset$  then
6 | this is the root cluster and broadcast the final list on the intracluster tree
7 end

```

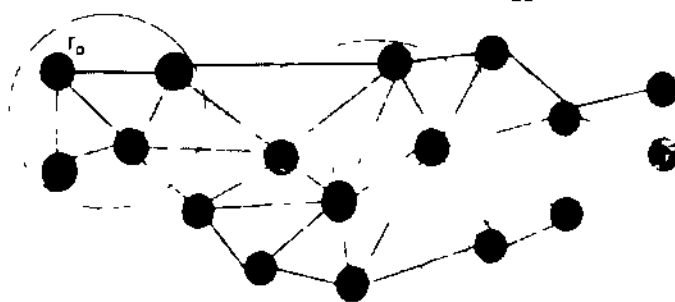


Figure 5.1 Cluster Creation

5.2.2.1 Example

Figure 5.1 is the example of the clustering algorithm. Which is start from the root node and produce clusters at the end of the algorithm. At node r_0 start the BFS tree. The first cluster is made of neighbors node because our basic condition becomes false at this stage. After first cluster the next cluster head is (suppose) r_1 . When the next cluster head is selected then start again the cluster creation procedure. After creation of two clusters we have applied the same technique on the remaining graph. We have also showed another example in section 6.3c, in which we can see that our scheme produces the clusters up to two hops.

Chapter 6

Results and Discussion

6.1 Correctness and Complexity

6.1.1 Correctness

For correctness, we will prove that the communication structure constructed by our algorithms is in fact, a tree. First, we must prove that our algorithm have produced a tree in cluster otherwise resultant network has a cycle somewhere. As a resultant, the network construct by our algorithm cannot be balanced in terms of load and latency.

Lemma 6.1.1. *the algorithm 5.2.1 is sparse in term of*

The cluster are disjoint i.e. $S \cap S' = \emptyset$ and

The graph $G(S)$ has intracluster edges not more than n^{1-k} (or the degree of the cluster S is n^{1-k})

Proof. Let suppose for contradiction that $\exists u \in S \cap S'$ then v belongs to Y and Z by definition this situation cause a contradiction, because the previous layer that is added to the cluster discard from the set of nodes and just those node added to the cluster which is not rejected.

This is clear from the sparsity condition that the edges touching the cluster S is less $\{6\}$

□

Lemma 6.1.2. *The algorithm 5.2.1 contain BFS tree in each cluster*

We will show by induction of t that the resultant tree of algorithm 3.1.10 is a BFS tree.

Basis

For $t = 1$ during the message *Msg* and *Leader node* the first BFS tree created with node 1.

Inductive step

Let this is true for $t - 1 = 1$. During round $t - 1$, Let f be a node that receives the message

Msg from the parent node f at distance $t - 2$. The node f received the message after distance $t - 1$. The node that received the message at distance $t - 2$ not changed his parent position neither delivered a message. And every node at distance $t - 1$ changed his parent as well as deliver the message. Thus, we can say that each node connects only those node which is at distance $t + 1$ e neighbors. In other words, each node has a parent node and any node not changed their parent and vice verse [9].

6.1.2 Complexity Analysis

Complexity analysis is essential for to check the performance of the algorithm. This is a function to check the speed of processing.

Lemma 6.1.3. *the communication and time complexity of the Cluster_creation procedure in the algorithm 5.2.1 is $O(|E| + |V| \log |V|)$ and $O(|V|)$ respectively.*

Proof. During cluster creation each cluster will call the cluster_creation procedure once, in at most $\log |V|$ pulse. At each pulse the message *Msg* and *count* is passed on each edge. The *count* and *Msg* message are passed through each edge and hence the complexity is $2 O(|E|)$. The bound for the diameter of the tree is $\log |V|$ and there is total $O(|V|)$ rounds so this yields $O(|V| \log |V|)$ complexity. So the communication cost is $O(|E| + |V| \log |V|)$. Each tree in the cluster has height $h \leq \log n$ and take h time units to reach the leaf nodes at h pulse. So the total time is $O(|V|)$ [6, 9]. \square

Lemma 6.1.4. *The communication and time complexity of next_leader procedure in algorithm 5.2.1 is $O(|V|^2)$ and $O(|V| \log |V|)$.*

Proof. The center of activity moves to next node by calling a subroutine *search_leader*. This subroutine broadcast *test* and convergecast *candidate* message to the leader of the existing tree leader. Also during *search_leader* subroutine, if the next layer is empty then a *RETRACT* or *Leader_node* message is sent to the chosen leader of the next cluster. These messages sended on the intracuster tree. So the communication is $O(|V|)$ and time is $O(\log |V|)$. The communication cost of the BFS procedure is $O(|V|)$ thus $O(|V|) \times O(|V|)$ yields total $O(|V|^2)$ communication complexity. The time complexity of DFS is $O(|V|)$. So $O(|V|) \times O(\log |V|)$ yields $O(|V| \log |V|)$ time complexity of the procedure *Search_leader*[13]. \square

Lemma 6.1.5. *The communication and time complexity of finding minimum edge in algorithm 5.2.1 is $O(|V|^2)$ and $O(|V| \log |V|)$.*

Proof. to find the minimum edge, the *preferred_link* procedure called at most once in the intracuster tree of the cluster. In general sense, the election process required $O(|V|)$ and $O(\log |V|)$ communication and time complexity. Since there is $|V|$ nodes and the election process is applied at most $(k - 1) |V|$ times. Thus the bound for total communication complexity is

$O(k|V|^{-2})$ As we know that the election process will perform on total on $|V|$ nodes so the preferred_Link procedure will take $O(|V| \log |V|)$ time [6-13]

□

The total time and communication complexity of our algorithm is $O(|V| \log |V| / \log k)$ and $O(K|V|^2)$

6.1.3 Comparison With Other Algorithms

In this section, we are going to compare our work with others our comparison emphasizes on cluster overlaps, node degree and radius of the network etc. Many papers described the cluster overlaps but not the node degree which causes the load of the network. During BFS procedure as we know that the algorithm ignores those edges which are repeated, i.e. a node received message *Msg* from more than one node (and select one node as father and ignore the other link). This procedure makes the load of the node low in other words the BFS produce a Tree. The radius of the intracluster tree can be controlled through the stopping condition i.e. $k \text{ (child nodes)} \geq \text{tree_node}$ where $k = \{1,2,3 \dots n\}$ and n is the total number of nodes. Table 6-1 shows some comparison with our algorithm. Overlapping of network mean one node belongs to more than two cluster and nodes receiving more and more messages from other cluster heads. This cause the communication complexity high. The algorithm of [16-17-40-41] have overlaps of the clusters which cause load on the network, but our algorithm guaranteed non-overlapping, each node is connected to only one cluster. Minimum radius mean, controlling the network latency. The latency of network become very high if the clusters based on one hop. In [16-17-28] the radius of the graph become very high as this algorithm is based on one hop clustering. We have tried to present such algorithm that is balanced in term of load and latency. In our presented algorithm, the radius is based on our condition. There may be a cluster based on neighbors and a cluster with two or three hops but our purpose is to balance the load and latency. The message size is considered fix as discussed in section 4-1-14.

Table 6-1 Comparison

Ref	Time	Message	commu- cation	deg(v)	Overlaps	Cluster Radius
Proposed scheme	$O(V \log V / \log k)$	Fix	$O(K V ^2)$	Minimum	No	Based on condition
Awerbuch et al [40]	$O(n^2)$	Fix		High if $k=1$ else low	Yes	Based on k
Moran et al [41]	$O(V + D \log V)$		$D \cdot k \cdot V $ $(D = F + V \log V)$	Maximum	Yes	Based on n
Lin et al [28]	$O(V)$			Maximum	No	One Hop
Chen et al [18]				Maximum	Yes	k Hop
Peleg et al [27]	$O(\log^2 n)$			$\sqrt{\log n}$	Yes	$\rightarrow \infty$
Bejnarczyk et al [17]	-			Maximum	Yes	One Hop
Marmali et al [37]	-			Minimum	Yes	k Hop
Aissa et al [16]	-			Maximum	Yes	$k+1$ Hop

In cluster architecture, the communication can be performed through cluster heads. If the number of clusters becomes large, this means that the communication overhead is maximum. Papers [16, 17, 28, 25] produce a maximum number of clusters and as a resultant, we face broadcasting overhead. We have minimized the number of clusters through our basic condition. Figure 6.1 shows the number of clusters in our proposed and previous literature. Our base paper for comparison is the Chen et al. [18] paper. Chen et al. [18] have minimized the number of clusters (for $k > 3$) but not concentrate on the degree of the node. Our proposed scheme becomes better than Chen et al. for $k \leq 2$ (see the red line in figure 6.1). The total number of clusters produced by Chen et al. and Aissa et al. algorithms are same because we have taken only twenty number of nodes (red and blue lines in figure 6.1). For a large graph the algorithm of Chen et al. is good as compare with Aissa et al.

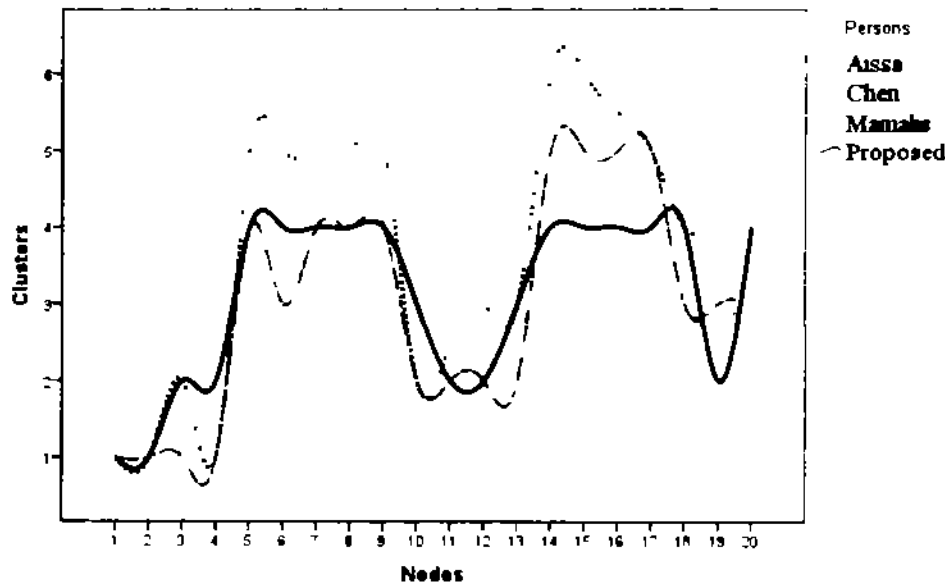


Figure 6.1. Cluster Comparison

In [16, 18] papers, the algorithm produces maximum clusters as compared with our proposed but we have noted that the number of degrees of node still not reduced. As [24] described that a node with a maximum degree will be in many clusters and each cluster head will send messages to it. In this case, the number of messages exchange become very high and as a resultant the communication complexity in intra-cluster and as well as inter-cluster tree become high. Chen et al. [18] reduced the number of clusters but not the cluster overlaps with other clusters. Mean that there is a common node between clusters. Figure 6.2 shows the degree and overlapping clusters. In Chen at al. papers, the cluster overlaps are high. Mamali et al. [25] have reduced the degree of the node as compared with us but in some cases the hope count increase as we can see in figure 6.3a. In degree comparison, we have tough competition with Mamali et al. (see the blue dotted line in 6.2)

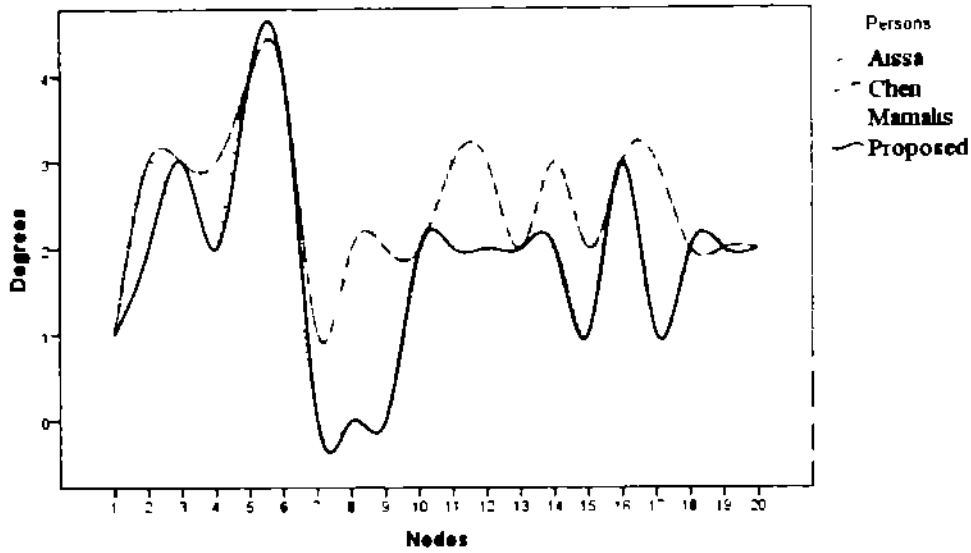


Figure 6.2 Degree Comparison

The general form of the cluster comparison with previous literature can be seen in figure 6.3. The cluster creation algorithm of Lin et al. [28] produced a number of clusters because this algorithm produces clusters of only neighbor nodes. The algorithm of Chen et al. [18] produce clusters with K-Hop nodes but not concentrate on degree control. When k is equal to one or two in Chen et al. algorithm then our proposed scheme become best but when k becomes equal to three our proposed scheme start a tough competition with it. To overcome the problem of k, we have also used k as to reduce the clusters in the graph. The intracluster communication is good from the [16, 28, 25] but not from [18]. The intercluster communication is same with [25] and good from others.

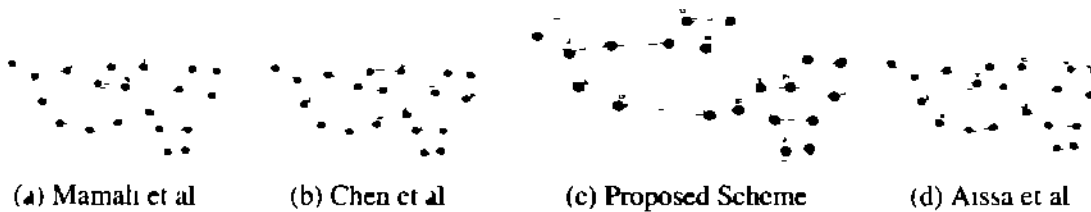


Figure 6.3: Comparison

Chapter 7

Conclusion and Future Work

In this thesis, we have considered the problem of an efficient structure of the network for communication among nodes. Each node is a part of a cluster and connected with cluster head. A network structure is efficient if there is a balance between number of hops (between the clusters), node degrees and no cluster overlaps. We have studied the problem related to load and latency. The clustering technique is an appropriate technique for load and latency balancing. We have ensured by lemmas that each cluster has a tree. This thesis presented an algorithm for the efficient partitioning of the nodes of an ad hoc wireless network into clusters with a cluster head. The cluster head selection is based on choosing the minimum id node in the neighbors. In this paper, we do not need information about the whole network because we are using the local information based model which is called Local Preserving Representation (shortly LP-Representation). In LP-Representation, each node only keeps information about their neighbors, itself and nothing else. This idea minimizes the space complexity of our scheme. In our scheme, we have used three main procedures, cluster creation, next leader procedure and finding the minimum outgoing edge. In cluster creation procedure we have created trees within clusters. As we know that each node have only one father node and there is no cycle in the tree. This technique minimizes the number of edges adjacent to a node which means that we are able to reduce the node degree through this technique. The communication and time complexity of cluster creation procedure is $O(|E| + |V| \log |V|)$ and $O(|V|)$. The next leader procedure is activated when the first cluster become completed. Next leader is chosen by sending a message to the new child nodes of the rejected layer through constructed tree which makes communication complexity low because we have removed extra edges. The communication and time complexity of next leader procedure is $O(|V|^2)$ and $O(|V| \log |V|)$. When the rejected layer is empty then go back to the root node by father nodes and find the minimum edges between the intracluster edges. When the minimum edge is selected then use this edge for communication. The communication and time complexity of finding minimum outgoing edge procedure is $O(|V|^2)$ and $O(|V| \log |V|)$. In our thesis we have made the intracluster as well as the intercluster communication low as possible. We have applied our algorithm on two graphs which

shows the clusters. A number of researchers have emphasized on load only while others have worked only on latency, however in our work, we have tried to make balance in both load and latency. We have compared our work with many papers. We have compared our scheme with different papers. From simulation results, it is evident that our work is balanced in term of node degree and overlaps. We will check our algorithm for performance on different models such as CONGEST MODEL and ASYNCH MODEL which is discussed in the thesis. As we have proposed a distributed algorithm for a communication structure, our next step will be a routing algorithm on proposed structure.

Glossary

Σ A finite Set of symbols | e deliver, compute, node and edges 6 7

δ A transition function such that $\delta : Q_i \times \Sigma \rightarrow Q_i$ 6 7, 37

Q_i A finite Set of States | e $Q_i = Q \times \bar{Q}$ 6 7

Q_0 initial state 6, 7

\bar{q}_i A finite set of links | e $v \rightarrow u$ and $u \rightarrow v$ 7

halt terminate the algorithm 37

inbuf the incoming message will be store in this buffer 7, 8, 36

outbuf the outgoing message will be store in this buffer 7, 8, 16, 36 37

Bibliography

- [1] G F Coulouris J Dollimore, and T Kindberg *Distributed systems concepts and design* pearson education, 2005
- [2] A Tanenbaum and M Van Steen *Distributed systems* Pearson Prentice Hall, 2007
- [3] S Banerjee and S Khuller, 'A clustering scheme for hierarchical control in multi-hop wireless networks,' in *INFOCOM Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings,IEEE* , vol 2, pp 1028–1037 2001
- [4] M Sipser, *Introduction to the Theory of Computation* Cengage Learning, 2012
- [5] H Attiya and J Welch, *Distributed computing fundamentals, simulations, and advanced topics* John Wiley & Sons, vol 19, 2004
- [6] D Peleg, 'Distributed computing' *SIAM Monographs on discrete mathematics and applications*, vol 5, 2000
- [7] F Kuhn, R Wattenhofer, and A Zollinger, "Asymptotically optimal geometric mobile ad-hoc routing," in *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications* ACM pp 24–33,2014
- [8] X Li, G Calinescu, and P-J Wan, 'Distributed construction of a planar spanner and routing for ad hoc wireless networks,' in *INFOCOM 2002 Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings IEEE* vol 3 pp 1268–1277,2002
- [9] J A Bondy and U S R Murty, *Graph theory with applications* Macmillan London, vol 6,1976
- [10] B N Clark, C J Colbourn, and D S Johnson, "Unit disk graphs" *Annals of Discrete Mathematics*, vol 48, pp 165–177, 1991
- [11] E W Weisstein, "Planar graph from "
- [12] T H Cormen, C E Leiserson R L Rivest C Stein *et al* *Introduction to algorithms* MIT press Cambridge, vol 2,2001
- [13] N A Lynch, *Distributed algorithms* Morgan Kaufmann, 1996

- [14] R B Muhammad, "Lecture notes Distributed algorithms,," kent State University USA
- [15] T S Rappaport, "Wireless communications principles and practice" 2002
- [16] M Aissa and A Belghith, "Quality of clustering in mobile ad hoc networks" *Procedia Computer Science*, vol 32, pp 245–252, 2014
- [17] W Bednarczyk and P Gajewski "An enhanced algorithm for manet clustering based on weighted parameters," *Universal Journal of Communications and Network*, vol 1, no 3, pp 88–94 2013
- [18] J S G Geng Chen, Fabian Garcia Nocetti and I Stojmenovic, "Connectivity based k-hop clustering in wireless networks," *Proceedings of the 35th Hawaii International Conference on System Sciences - 2013*
- [19] P Bose, P Morin, I Stojmenović, and J Urrutia "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless networks*, vol 7, no 6, pp 609–616, 2001
- [20] M Dyabi A Hajami, and H Allali, "A new manets clustering algorithm based on nodes performances" in *Next Generation Networks and Services (NGNS), 2014 Fifth International Conference on* IEEE, pp 22–29 2014
- [21] R B Muhammad, "A distributed graph algorithm for geometric routing in ad hoc wireless networks" *Journal of Networks*, vol 2, no 6, pp 50–57, 2007
- [22] E Kranakis, H Singh, and J Urrutia, "Compass routing on geometric networks" in *Proc 11th Canadian Conference on Computational Geometry* Citeseer, 1999
- [23] B Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems" in *Proceedings of the nineteenth annual ACM symposium on Theory of computing* ACM, pp 230–240, 1987
- [24] S Banerjee and S khuller, "A clustering scheme for hierarchical control in multi-hop wireless networks," *IEEE Infocom 2001, Vol 2, 1028-1037* vol 2, no 1 pp 226–237, 2001
- [25] B Mamalis, D Gavalast, C Konstantopoulos, and G Pantziou, "Clustering in wireless sensor networks," *RFID and Sensor Networks Architectures Protocols Security and Integration*, Y Zhang, LT Yang, J Chen, eds, Pag 324-353, 2009
- [26] B Awerbuch, "Complexity of network synchronization," *Journal of the ACM (JACM)* vol 32, no 4, pp 804–823, 1985
- [27] B Awerbuch and D Peleg "Sparse partitions," in *Foundations of Computer Science 1990 Proceedings, 31st Annual Symposium on IEEE*, pp 503–513, 1990
- [28] C R Lin and M Gerla, "Adaptive clustering for mobile wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol 15, no 7, pp 1265–1275, 1997

- [29] D Yu, Q-S Hua, Y Wang, J Yu and F Lau, "Efficient distributed multiple-message broadcasting in unstructured wireless networks," in *INFOCOM, Proceedings* IEEE pp 2427–2435 2013
- [30] A Kesselman and D Kowalski, "Fast distributed algorithm for convergecast in ad hoc geometric radio networks," in *Wireless On-demand Network Systems and Services WONS Second Annual Conference on* IEEE pp 119–124 2014
- [31] B Awerbuch, "A new distributed depth-first-search algorithm," *Information Processing Letters*, vol 20, no 3 pp 147–150, 1985
- [32] R G Gallager, P A Humblet, and P M Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol 5, no 1, pp 66–77 1983
- [33] P L Rice, A G Longley K Norton, and A Barsis, "Transmission loss predictions for tropospheric communication circuits, volume 1," DTIC Document, Tech Rep , 1967
- [34] Y Okumura, E Ohmori, T Kawano, and K Fukuda "Field strength and its variability in vhf and uhf land-mobile radio service." *Rev Elec Commun Lab*, vol 16, no 9 pp 825–73, 1968
- [35] M Hatay "Empirical formula for propagation loss in land mobile radio services" *Antennas and Propagation Magazine, IEEE Transactions on* vol 29 no 3 pp 317–325 1980
- [36] T K Sarkar Z Ji, K Kim A Medouri, and M Salazar-Palma, "A survey of various propagation models for mobile communication" *Antennas and Propagation Magazine IEEE*, vol 45, no 3, pp 51–82, 2003
- [37] T S Rappaport, "The wireless revolution," *IEEE Communications Magazine*, vol 29 no 11, pp 61–71, 1991
- [38] D Cox, R Murray, and A Norris, "Measurements of 800-mhz radio transmission into buildings with metallic walls" *Bell System technical journal* vol 62 no 9 pp 2695–2717, 1983
- [39] W Bednarczyk and P Gajewski, "An enhanced algorithm for manet clustering based on weighted parameters," *Universal Journal of Communications and Network*, vol 1, no 3, pp 88–94, 2013
- [40] B Awerbuch, B Berger, L Cowen, and D Peleg, "Fast distributed network decompositions and covers," *Journal of Parallel and Distributed Computing*, vol 39 no 2, pp 105–114, 1996
- [41] S Moran and S Snir, "Simple and efficient network decomposition and synchronization," *Theoretical Computer Science*, vol 243 no 1, pp 217–241 2000