Distributed Network for Broadcast Communication

MS Thesis

By:

Aneela Javed

680-FBAS/MSCS/S12

Supervisor:

Dr. Rashid Bin Muhammad

Department of Computer Science & Software Engineering

International Islamic University, Islamabad

March 24, 2016

Abstract

As the application of wireless communication technology, internet and the rapid development of electronic technology, people demands of data services in the wireless environment have become very urgent. More and more users hope to access the Internet by wireless and enjoy fast network services. The problem of constructing efficient structure for broadcast communication in wireless network is investigated. A simple distributed solution for balancing load and latency of network in synchronous model is provided. The network is locally decomposed into small connected induced subgraphs. The problem of constructing efficient network structure is motivated by its role in scheduling problem, optimal route selection for broadcasting, load balancing etc. Focusing on constructing efficient structure in a localized manner the set of nodes that belongs to the same class are clustered in order to decrease overall time and communication complexity. By using efficient MIS technique, we proposed fully localized algorithm for constructing a convenient structure for wireless communication. The algorithm requires total $O(n)$ messages whereas the time complexity is proportional to the number of clusters created.

Certificate of Originality

This is to certify that we have read the thesis submitted by, Aneela Javed registration number 680/FBAS/MSCS/S-12. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the degree of MS in Computer Science.

Committee Members

External Examiner

Professor Dr. Muhammad Abdul Qadir

Dean Faculty of Computing

Capital University of Science and Technology

Internal Examiner

Ms. Umara Zahid

Lecturer

Department of Computer Science and Software Engineering.

International Islamic University Islamabad

Supervisor

Dr. Rashid Bin Muhammad

Assistant Professor

Department of Computer Science and Software Engineering.

International Islamic University Islamabad

Contents

List of Figures

List of Algorithms

List of Tables

ACKNOWLEDGMENTS

In the name of Allah, the most Gracious and, the Most Merciful.

Though only my name appears on the cover of this thesis, a great many people have contributed to its production. I owe my gratitude to all those people who have made this work possible and because of whom my masters experience has been one that I will cherish forever.

My deepest gratitude is to my advisor, Dr. Rashid Bin Muhammad. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. His patience and support helped me overcome many crisis situations and finish this thesis.

I would also like to thank all my teachers, staff, lab assistants, coordinators who have made our department a very special place over all those years.

Most importantly, none of this would have been possible without the love and patience of my family. My lovely family to whom this thesis is dedicated to, has been a constant source of love, concern, support and strength all these years. I would like to express my heart-felt gratitude to my parents Javed Iqbal and Riffat Javed, my brothers Saqib and Asif, my sisters-in-law Nosheen Saqib and Maryam Asif for their love, support and encourgment.

I am also grateful to all my friends who encouraged me through these difficult years. My dear friends Sadaf, Mahwish and Sadia who always supported me during my stressful and difficult moments.

1 Introduction

This thesis deals with the problem of constructing locality sensitive distributed algorithm for construction of sparse network for broadcasting in wireless ad hoc networks. The idea of this problem is inspired by [1, 2, 3, 7]. The study of distributed network algorithms is motivated by its central role in network design. Network load and latency are the two basic parameters of concern for designing a sparse network. Our problem deals with balancing network load and latency since there is a tradeoff between these two parameters.

1.1 Motivation

The problem of constructing distributed sparse network for broadcasting is a well-known and oldest problem in network design like other combinatorial optimization problems such as range assignment problem, maximum flow problem etc. Researchers have put a lot of research effort into this problem because of wide range of applicability such as routing, online searching of mobile users, task low load scheduling etc.

The network is modeled, as a graph in which vertices represent set of nodes and edges represent communication links between these nodes. Formally, a geometric graph is defined as $G = (V, E, w)$, where $V(G)$ is the set of vertices (or nodes) of the graph, $E(G)$ is the set of edges and $w : V \rightarrow R^+$ is a weight function associated to each node of graph. A vertex u is the neighbor of any other vertex v in graph if they are connected by an edge. A tree is a non-cyclic graph. We are interested in several types of tree structures specially, a minimum spanning tree and a tree with the property of shortest path. There are no. of techniques through which we can extract tree from the graph like, breadth first search that gives BFT tree, and Depth first search that produces DFS tree. These are efficient structures for exchanging information and performing other operations in a wireless network. The subgraph is also an important part of this discussion, a subgraph U of any graph G is by definition a graph whose set of vertices and set of edges are all subsets of G, in detail defined in section 3. In next section we discussed parallel and distributed models of network computing.

1.2 Parallel and Distributed Computing

A parallel computing is the type of computing in which many computations perform simultaneously, also we can state it as a system that works on a divide and conquer approach. Processors

in a parallel computing use shared memory for inter process communication; more then single processors can run independently but share the common memory resources. If any particular processor made change in memory location, all other processor can see the change made by it. In a shared common memory model each process accesses a shared address space that opens up a lot of concurrency issues. In the message passing model an application execute as a collection of independent processes, each having its own local memory. Shared memory and message passing are two opposing communication models for parallel multiprocessor architecture.

Distributed computing systems are distinguished by their structure, a classical distributed system contains high number of intercommunicating processors that each execute their own programs but that are affected by receiving messages. The distributing systems range from simple systems in which one client talks to one server to huge unstructured networks like the internet [21]. In a typical distributed environment each processor works with its own local memory and processor communicate through message passing. Tasks are divided among these processors to minimize the total running time. Parallel and distributed systems have a lot of overlap because the processors in a typical distributed systems runs concurrently parallel. The major difference among them is the inter-process communication, typically for parallel computing shared memory model is used, whereas the distributed computing algorithms work on message passing model.

Distributed algorithms are designed for such type of systems which run independently and each with specific amount of information. Here we discuss models namely, synchronous and asynchronous of a distributed computing and then several distributed algorithms with respect to these models. These models are used for specification of the problem to be solved and also for the verification of algorithm correctness.

Now we shall focus on a message passing synchronous and asynchronous model.

1.3 Message Passing Model

In a distributed computing each component or processor is located at geographical distance having only local memory. Processors communicate by sending and receiving messages. The message is send to a receiver, which executes the message, and then replies by sending message. In turns, the reply may trigger a further request, which may leads to a succeeding response, and so forth. The exchange of information between processes demands "interactive action" to be performed by each process.

1.3.1 Formal definition and Transition function

Formal definition and Transition function

A system consist of a finite set of p processors $p_0, \ldots p_{n-1}$, i is the position of processor p_i,a set of finite states Q_i in addition there are two components that each state of p_i consists of an outbuf denoted as, $outbuf_i[l]$ and an inbuf denoted as, $inbuf_i[l]$ for every $l, 1 \leq l \leq r$ which is a finite set of messages. The $outbuf_i[l]$ contains messages that p_i has sent to its neighbors over its lth link but not have yet been delivered and $inbuf_i[l]$ contains messages that have been delivered to p_i on lth link but yet not processed with an internal computation by p_i. A distinguished subset of initial state.

For the processors p_i transition function it takes as input a value for the approachable state of p_i, it generates as output a value for the approachable state of p_i in which each $inbuf_i[l]$ is empty. Each produces a message from the set of messages: which is delivered to the neighbors on the incident links. This step do not interrupt with the previous step. All the messages that are waiting to be delivered to p_i are processed in each step and eventually it causes state change.

Each processor is initially in an initial state of q_i, there are few events in message passing model that occurs, that are computational event denoted as $comp(i)$ of p_i in which transition function is applied to reachable states. And the other is delivery event, denoted as $del(i, j, m)$ represnts the delivery of message from p_i to p_j. Number of these configuration cause an execution of event.[20]

Further we investigate the asynchronous and synchronous type of message passing model.

1.3.2 Synchronous Message Passing Model

Synchronous Message Passing

In a synchronous message passing model operations are accomplished under the single central unit control of a fixed rate clock signal or several clocks and there is no timing question. Systems execute in a succeeding manner. The sequence of configuration is divided into rounds, and in each round, every processor can send a message to each neighbor, the messages are delivered, and every processor computes based on messages frequently received. The basic assumptions that are considered in this model are, every processor has same speed. Each processor takes same amount of time to perform same task. This is the reason that in this model algorithms are much easier to design but at the same time it makes them less resistant to real world timing oddities. These algorithms are much easier to transform in to asynchronous algorithms by applying synchronizer. [20]

1.3.3 Asynchronous Message Passing Model

Asynchronous Message Passing

Asynchronous systems are event based instead of a fixed clock time. There is no predetermined upper bound on how long it takes for a message to be delivered or how much time pass by in between consecutive steps of a processor. A procedure is called permissible if there are infinitely many computation steps for each process, and every message is finally delivered. These are the fairness conditions [20].

Example: Client Server Computing

Client-Server computing is the example of asynchronous model, which describes the pattern of work sharing. It is the most common model of distributed computing. the first process, the server manages some resources for example, the service is to enhance or upgrade the solution. The server once started, runs permanently over a long period, then it is passive in that it stops the process and waits until client sent request. The client is the process that sends request in order to fulfill its requirements. It sends request according to its requirement to the services. Upon receiving this request the server awakes checks the query and sent back the desired result to the client. After that server again stops its process and waits for the other request from clients. in order to achieve parallelism in this process the client manages multi threading so that processor shift to another request while waiting for a previous request. by achieving this parallelism the performance can be improved. [27]

We have studied the fundamental problems related to distributed networks and locality sensitivity distributed computing, More specifically we target the problems like broadcasting, sparse partitioning. Broadcasting is a major tool for solving many other problems in distributed algorithms.

Previous techniques for achieving different global network control and management functions such as message routing, broadcasting are relay on requiring all entities to keep track full information of each and every state of network. As distributed networks gets larger, many tasks become more complicated, expensive and resource consuming. This is also experienced that universal information is not always necessary many tasks can be performed with limited information i.e. processors only required its immediate neighborhood information. Certain tasks are limited in nature that is they need information about a particular region of the network. Furthermore it is necessary that the cost of the task be corresponding to its locality level. For a distributed network we have two extreme cases, one is when every node has information of every other node or has global information which is very non practical though we get high accuracy but this is impractical and expensive,

on the other hand we have worst case in which any node knows nothing about network which is also the other extreme case. Our point of view is to utilize "locality of reference" i.e. carrying out these tasks needs processors to know more about their neighbors and less about the rest of the network [2]. This idea is less expensive and less resource consuming as well, so we compromise on accuracy and try to get improvement in the tasks. We are interested in locality preserving network representation because it substitutes the complete representation of the network and permits us to set the less amount of information and also reduces the computation and memory requirement. We have acquired this idea from [2]

1.4 Latency and Load trade-off in Network Construction

We narrow down the problem by defining the key factors that are highly considered while constructing a network. These two major parameters are network load and latency, latency of the network is defined as the total time it takes for a packet to cross the network from a node that propagated that packet to the intended node. Delay could be considered as a synonym of latency. Low latency guarantees low time complexity. On the other hand load refers to the amount of data (traffic) being carried by the network. The major problem is, these two parameters are inherently conflicting, that is improving one of them results dropping the other parameter. Therefore this tradeoff motivates us to analyze and design such a protocol that provides an approximate solution for balancing these factors. [1, 2] The goal is to design such a protocol that uses minimum amount of energy in terms of load and latency and provides an optimal structure for broadcast communication.

1.5 Proposed Approach Overview

In the light of above discussion we determined that it is convenient to develop a system based on hierarchical or clustered representation of networks. This technique permits processors to maintain only limited information concerning the state of the network. Clustering is one of the fastest divide and conquer technique for solving this problem. We investigate some of the techniques and benefits for using this approach. We partition the network into clusters and try to balance the locality and sparsity factor. Our inspiration is [2, 3, 7, 20, 27, 39]. We used efficient Maximal independent set (MIS) construction technique to solve clustering because the problem of constructing MIS is technically same as clustering.

This thesis considers the classical, log distance path loss model of communication that is a transmitter-receiver communication model. These models are studied in detail in next chapters.

1.6 Structure of the thesis

The section 2 will discuss the model of computations. The section 3 and 4 will discuss preliminaries and literature survey, respectively. Section 5 will discuss the problem and the main contribution of chapter 6 is the approximate solution of the problem, finally section 7 discuss complexity analysis, results of the algorithm.

2 Wireless Computational Models

In this section we present some of the computational models that we used in solving our problem in wireless networks. To study algorithmic issues we must construct the appropriate model of system and computation.These models represent the protocol features that are common to both hardware and software. For any problem we have number of computational models with their advantages and drawbacks. In our work we used and study distributed synchronous and asynchronous model for designing algorithms and for complexity analysis. We also used log distance path loss model of propagation for wireless networking.

There is lot of research done in development of various analytical and empirical based models to measure ratio propagation in various frequency bands and for different types of environments. These models of signal propagation are designed to estimate the average power of the received signal from a certain distance from the transmitter, and also the variation in intensity of the signal. These models are also used to predict the SNR which is known as signal to noise ratio (the estimate of signal in presence of any distraction or disturbance).

The Reflection, diffraction and scattering are the three factors in propagation that impacts the wireless communication. Reflection is cased by any large body on which wave imping,s and wave length is smaller then this body e.g. reflection from ground and buildings. These reflections often effect signal in many ways. The other case is diffraction, it happens when the ratio path between sender and receiver is disturbed by an impenetrable body and a surface with sharp irregularities. Finally, scattering happens when the ratio link have objects whose volume are on the order of the wave length or less of the propagation wave and also when the numbers of distractions are large.[14]

Receiving power of any receiver depends on the transmitting power of the transmitter and the path loss therefore we can write it as;

$$P_r = P_t / P_L(u, v)$$

Where $Pr \geq \beta$ where $\beta = 1$ is a threshold for the propagation. P_L is path loss, P_r is receiving power of the receiver and P_t is the transmitting power of transmitter, u represents the transmitter and v is used for receiver.

2.1 Path loss models

Path loss is defined as reduction in signal strength.There are no. of models designed for coverage prediction and distraction analysis for designing of wireless communication systems. Both analyti-

cal and empirical based propagation models highlight that average received signal power decreases logarithmically with distance. Log-distance model is an indoor empirical propagation model.

2.1.1 Free-Space Propagation model

This propagation model is the base of all models, which is used when there are no reflections or multipath between the transmissions.[15] Mathematically this model is;

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

Here $P_r(d)$ is receivers receiving power of signal at a distance d form t. Transmitted power is denoted by P_t . G_t and G_r presents the transmitter antenna gain and the receiver antenna gain respectively. d is the T-R separation distance. The system loss factor is represented by L and λ is the wave length. By substituting $(G_t, G_r, \lambda^2, 4\pi^2, L)$ with c as a constant factors we have

$$P_r(d) = cf.P_t/d^2$$

where f is the free space because $P_r \geq 1$ so $cf.P_t/d^2 \geq 1$ and $d \geq \sqrt{cfP_t}$. Eventually we got a formula to find a radius of a disk which is used to calculate a disk radius for this particular case which is used to determine the range of the disk or more specifically node.[16, 17]

Table 1: Path loss exponent

Environment	Path loss exponent, n
Free space	2
Urban area cellular radio	2.7 to 3.5
Shadowed urban cellular radio	3 to 5
In building line-of sight	1.6 to 1.8
Obstructed in building	4 to 6
Obstructed in factories	2 to 3

2.1.2 Two ray propagation model

As the name indicates it's the two ray propagation model. To understand this, suppose we have a transmitter antenna u and at a certain distance receiving antenna v. The signal from transmitter to receiver propagates through two rays, one through direct link and other is through the reflection from any other medium supposing the smooth earth surface. Therefore we have;

$$P_r(d) = P_t G_t G_r ht^2 hr^2 / d^4 . d^4$$

Hence, again from this we got a radius of a disk with two ray propagation model. By combining these two models we got more general scheme known as log distance path loss model.

2.1.3 Log distance path loss model

Log distance path loss model is a very popular logarithmic model that is based on a linear dependence between the path loss in decibels (dB a logarithmic unit that is used to present a ratio) and the logarithm of the distance between the transmitter and the receiver.

$$P_r(d) \propto P_t/d^{\alpha}$$

Where α is a distance power gradient that varies from environment to environment for example in free space, n is equal to 2 and in presence of any obstetrical, value of n will be larger. The path loss is measured by a free space path loss formula or through field measurement at distance d_0. In huge coverage mobile systems 1 km reference distance are commonly used, on the other hand in microcellular system, much smaller distance like 100 m or 1 m are used. The base of all of these models is free-area propagation model which is used to estimate received signal power where the transmitter and receiver have clear, noninterfereing path between them. According to various studies log-distance model is accurate and simple to use. Table 1 shows the path loss exponent based on different environment. [18, 19]

2.2 Role of models

We are continuously focusing on different models in our work it is due to the reason that computational models specify the problem and provide mechanism for verification of the correctness of proposed algorithm. As an example suppose log distance model is used to specify the transmitter receiver power for communicating in wireless sensor network. Another model which is unit disk graph also provides the connectivity range between nodes. In this section we avoid discussing synchronous and asynchronous model of distributed computing because they are being discussed throughout the thesis.

3 Prelimenaries

In this chapter, we look at the basic definitions, lemmas in a very broad way to set the stage of next chapters. The goal of this chapter is to provide a reasonably good grasp of the terminologies of the work. Most of the definitions are taken from the standard graph theory book [2, 10]

3.1 Distance in Graph

The distance of two vertices in graph which is denoted by $dist_G(u, v)$ where u, v are the two vertices of graph is defined as the length of the shortest path between these two vertices. The length of the shortest path represents the number of links that exist when we travel from u, to v. In a graph G where $G = (V, E, w)$, the weight function w often represent the length of links.

Similarly if there are two set of vertices U, V in graph, the distance between set U and V denoted as, $dist(U, V)$ is defined as the minimal distance $dist(u, v)$ in any two vertices u belongs to set U and v belongs to set V.

$$distG(U, V) = min\{dist_G(u, v) | u \in U, v \in V\}.$$

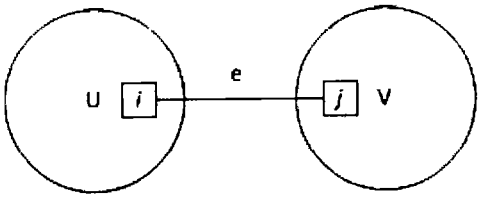


Figure 1: The links e = (u,v) and the sets it joins
[2]

3.2 Vertex Connectivity

The connectivity of vertex v in G is recognized by $deg(v)$ which is in-fact total number of links incident on v. Formally;

$$deg(v) = |\{u \in V : (v, w) \in E\}|$$

Minimum and maximum connectivity is denoted by, $\delta(G)$ and $\Delta(G)$ respectively.

3.3 Diameter and Radius

In a connected graph the eccentricity of any vertex v is computed by tracking the farthest vertex from v in graph. The radius in the graph is the minimum eccentricity whereas the diameter is the maximum eccentricity.

Formally;

$$Diam(G) i.e. max_{u,v \in V}(dist_G(u,v))$$

$$Rad(G) i.e. min_{v \in V}(Rad(v,G))$$

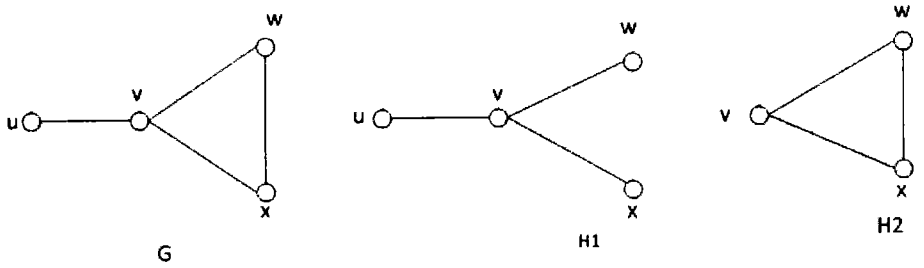
For any graph its center is the node realizing the radius of graph such that $Rad(v,G) = Rad(G))$. There is another very important property about radius and diameter i.e. radius of graph is less then or equal to the diameter of graph which is less then or equal to the two times diameter of graph.

3.4 Isomorphic Graphs

The isomorphic graphs are similar in their structure which means that have correspondence in all ways i.e. link to link and vertex to vertex.

Suppose $S_1 = (V_1, L_1)$ and $S_2 = (V_2, L_2)$ are two graphs. An isomorphism from S_1 to S_2 is a correspondence (bijection) $\phi : V_1 \rightarrow V_2$ such that $v, u \in L_1$ if and only if $\phi(v), \phi(u) \in L_2$

An isomorphism of a graph to itself is called automophism or symmetry. Figure 2 shows an isomorphic graphs, A subgraph of a graph G is a graph H such that $V_H \subset V_G$ and $L_H \subset L_G$. Usually any graph isomorphic to subgraph of G is also said to be a subgraph of G.



A spanning subgraph H1 and an induced subgraph H2

Figure 3: Subgraphs

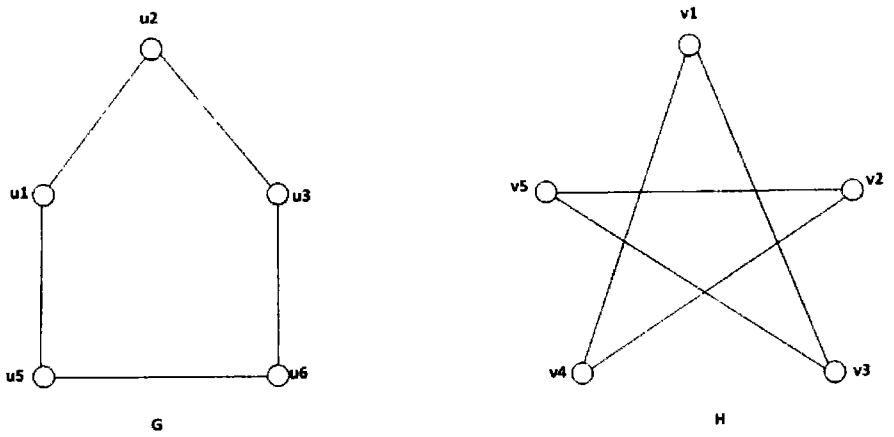


Figure 2: Isomorphic graphs

Further we discuss some kinds of subgraphs because our main focus of work is to extract a particular subgraph in the network to perform routing.

3.5 Induced and Spanning Subgraph

The induced subgraph h consists of few vertices of graph G and all the edges that connects these vertices in the original graph G, thats why is is called connected.

Formally;

$V(G(W)) = W and E(G(W)) = \{e \in E(G) |$ the endpoints of edges e are in $W\}$

A spanning subgraph contains all the vertices of original graph but with few number of edges.

In figure 3, H1 is a spanning subgraph of G but its not an induced subgraph, and H2 is an induced subgraph but not a spanning subgraph. A graph has more than one spanning trees, a minimum cost spanning tree is a spanning tree with cost less than or equal to every other spanning tree of graph. Formally we define Minimum spanning tree as;

A graph G such that $G = (V, E, c)$, the minimum cost spanning tree of G is a spanning tree T minimizing $c(T)$, where $c(G') = \sum_{e \in G'} c_e$ for any sub graph $G' \subseteq G$. Finding the MST is among the major problems in network design. In a mobile communication network an MST is used as a communication tree to minimize energy consumption.

3.6 Predecessor Graph

As name indicated the predecessor subgraph is basically a connected tree with the property of shortest path.

Formally,

For a graph $G = (V, E)$ with root s, a predecessor subgraph is: $G\pi = (V\pi, E\pi)$ where, $V\pi = \{v \in V : v.\pi \neq NIL\} \cup \{s\}$ And $E\pi = \{(v.\pi, v) : v \in V\pi - \{s\}\}$.

Considering the ring network, it is quite obvious that if all the nodes in the given graph are identical, then the computation cannot be perform at all on it. Following is the important lemma for the impossibility result of identical processors.

3.7 Lemma: Impossibility result

Let A be a system of n nodes where $n > 1$, arranged in a bidirectional ring. If all the nodes in A are identical, then A doesn't solve the leader election problem.

Proof. Suppose that there is such a system A that solves the leader election problem. We obtain a contradiction.. WOLG. We suppose that each process of A has exactly one start state. With this assumption, A has exactly one execution. So consider the unique execution of A it is straight forward to verify, by induction on the number r of rounds that immediately after r rounds all the nodes are in identical states. Therefore if any node ever reaches a state where its status is leader, then all the nodes in A also reach such a state at the same time. But this violates the uniqueness requirements.

Lemma says that if we wish to solve the leader election problem then we must break the symmetry somehow. A reasonable assumption derived from this statement is that the nodes in the given graph

are identical except for an ID. [11] This is the assumption we make in the rest of this thesis.

3.8 Algorithm GHS

In this section we discussed the most important distributed algorithm that finds minimum cost spanning structure of the network. It is the best known algorithm till now. The advantages of this technique are discussed in detail under literature survey. Here we just formally discussed the steps of algorithm.

Frontier And Blue Edges

For solving the distributed MST problem it basically works on two types of edges, a frontier edge i.e. a non-tree edge with one endpoint in tree, called its tree endpoint and one endpoint not in tree, its non-tree endpoint. We grow tree by repeatedly choosing a frontier edge and adding it to the tree.

This algorithm deals with fragments which is a subgraph of tree, it basically creates fragment and merge them by minimum weight outgoing edge. The minimum weight outgoing edge from fragment is called the blue edge.

Now we discussed the algorithm steps for constructing distributed minimum spanning tree. This algorithm has significant importance in our work because we can choose minimum weight outgoing edge between clusters through this algorithm. The detail description is presented in review of literature, here we formally written the basic steps of the algorithm.

Input : G = (V, E)
Output: MST T
Initially each vertex is the root of its own fragment. We proceed in phases:
repeat
All nodes learn the fragment IDs of their neighbors.
The root of each fragment uses flooding in its fragment to determine the blue edge $b = (u, v)$ of the fragment.
The root sends a message to node u; while forwarding the message on the path from the root to node u all parent-child relations are inverted (such that u is the new temporary root of the fragment)
node u sends a merge request over the blue edge $b = (u, v)$.
if *node v also sent a merge request over the same blue edge $b = (v, u)$* **then**
| either u or v (whichever has the smaller ID) is the new fragment root
| the blue edge b is directed accordingly
end
else
| node v is the new parent of node u
end
the newly elected root node informs all nodes in its fragment (again using flooding) about its identity
until all nodes are in the same fragment (i.e., there is no outgoing edge)

Algorithm 3.1: [Gallager–Humblet–Spira (GHS)]

4 A Survey of Prior Literature

In this chapter we look at the prior literature to describe the background and the base of our problem. In the first section we discussed the complexity measures of distributed algorithms and the basic proof methods. We also discussed basic communication operations to solve the problem of sparse partitioning, clustering. And the base literature is surveyed to get the strong grasp of problem. This chapter covers the base of thesis and also settles the ground for next chapters.

4.1 Introduction

To simplify computational processes and to solve different problems in the network it is essential to first set up various structures. For instance it is possible to construct spanning tree on the network. If all the processors know this spanning tree, it helps to perform various tasks related to communication like broadcasting, leader election, partitioning etc. In distributed environment having the local information means knowledge of tree. i.e. a node knows it self and its one hop neighbors.
In many cases we require subgraph to perform function on it. The problems of finding a breadth first search tree, shortest path or minimum spanning tree are motivated by the need to build structures suitable for supporting efficient communication. We discussed these structures in detail; furthermore we described protocols for propagating information in a network that are broadcast convergecast, downcast and upcast as well as computation complexities of performing these operations.

Our main focus of studying these tools is building the base for solving the problem of sparse partitioning of the network. Among all these we majorly focus on communication primitives and the computational complexity of these operations.

4.2 Complexity Measures for Distributed Algorithms

For distributed algorithms the time and communication complexity is considered high, although the space complexity is still considerable but for the distributed algorithms our major goal is to minimize total communication and time used by any algorithm.

4.2.1 Time Computation

In generally the time of any algorithm is measured by the total steps an algorithm takes from initial step to the last execution. In the distributed computing we distinguish synchronous and asynchronous time because these two are the different models of computation and hence requires different time.

Time of Synchronous algorithm

In the synchronous model the processors executes under the fixed clock rate therefore the time computation of a synchronous algorithm is defined as the number of pulses generated during the execution of algorithm on the network in the worst case, from the time the first processor starts the execution of algorithm until the last processor has finished execution.

Time of Asynchronous algorithm

The asynchronous model is non-deterministic that is we can not surely determine the time in this model rather it is event based model. The time of such algorithms are measured by computing the number of time units from the start of the execution of algorithm on network until its completion in the worst case. Assuming that each message incurs a delay of at most one time unit. [2].

4.2.2 Communication Complexity

The communication complexity of both synchronous and asynchronous algorithms are measured by the total amount of messages shared during all admissible execution of algorithm. The length of the message is always $O(\log n)$ bits.

we also consider the space required by algorithm therefore we define it as the maximum memory bits an algorithm takes in the worst case execution.

4.2.3 Proof Methods and Complexity Bounds

In order to prove that our proposed algorithm is correct and reasonable we have to verify its correctness through some method. For distributed algorithms we have two well known verification methods, simulation and invariant insertion. Invariant insertion are proved by mathematical induction technique on the number of steps of the algorithm execution. Simulation is also generally proved by the same procedure on the number of completed rounds. For simulation method the goal is to show that one procedure A implements another procedure B, producing the same input and

output behavior. This simulation between A and B is expressed by assertion relating the states of A and B, when the both procedures are started on the same inputs for the same amount of rounds. [3, 20]

In this section we discussed some bounds on complexity and their notation. The standard $Big\text{–}Oh$, $Omega$ notation. We always take the worst case scenario of our algorithm specifically upper bounds are global by default. Lets say for example, our algorithm δ has time complexity $O(\log n)$ on some class of graphs, it means following:

There is a static $t > 0$ *such that for every particular class of graph* G, $Time(\delta, G) \leq t(\log n)$, *or more strictly for every permitted input for* δ, *in every execution scenario of* δ *on* G, *our algorithm will ends within time at most* $t(\log n)$.

Comparatively in case of lower bounds of worst case, when we write that our algorithm δ has time complexity $\Omega(n)$ on any graph it states that:

There exists a constant $t > 0$ *such that for every* $n \geq 1$, *their exist an n vertex graph* G *such that* $Time(\delta, G) \geq t(n)$, *i.e. the algorithm ends after time of at least* tn.

We also discuss universal lower bound for every graph in the particular given class, for example if we say that the algorithm δ has time complexity $\Omega(n)$ on every graph means:

There is a constant $t > 0$ *such that for every n vertex graph the* $Time(\delta, G) \geq t(n)$,*i.e. in which the algorithm ends after time of at least* tn.

We conclude the difference in bounds is, that the lower bounds only implies the presence of some specific network in the class under discussion for which the bound holds on the other hand the universal lower bound must exist for every network in the class.[2]

4.3 Communication Primitives

The communication primitives works as the basic building block for distributed algorithms. These operations provide mechanism to propagate information among nodes and help to create channels for network. We discuss basic communication operations like broadcast, convergecast, downcast and upcast on a tree, and complexities of performing these tasks.

4.3.1 Simple Broadcast: Automaton

When a node wishes to propagate information to all other nodes of the network it uses broadcast mechanism. Its the basic building block of the network. It is very important to get inside the

functionality of broadcasting a message through a single initiator to the source. Following is the simple automaton code that broadcast a message M.

We have input, output, set of states and a transition function.

Input	**Output**
$bcast(m)_i, m \in M, 1 \leq i \leq n$	$receive(m)_{i,j}, m \in M 1 \leq i, j \leq n$

States

For every $i, j, 1 \leq i, j \leq n$:
Queue i, j, a FIFO queue of elements of M, initially empty

Moving Function

$bcast(m)_i$
Effect
For all j do
add m to queue(i, j)

$recieve(m)_{i,j}$
Precondition
m is first on queue i, j
Effect
remove first element of queue(i, j)

Tasks

For every i, j : $Receive(m)_{i,j} : m \in M$

Algorithm 4.1: bcast Automaton

4.3.2 Broadcast on Spanning tree

A spanning tree can be used to broadcast information as well as broadcasting itself grow a spanning tree rooted at some node i.e. we can use flooding to construct a spanning tree. Following is the simplest procedure of broadcasting on a given spanning tree rooted at P_r. The procedure broadcast initiate by the root of the tree and each node when receives the message from its ancestor forwards it on tree edges to its decedents.

The root propagate message M to all of its children in first round and then each processor upon receiving M from predecessor forwards it on the tree edges to its successor. When a spanning tree

Input : A Spanning tree T, with a distinguish root node P_r, a message M, We have, $Parent_i$ that holds index of i or nil in case of P_r, $children_i$ that holds set of indices of i, and a $terminated_i$ which shows that i is in terminated state

Output: Broadcast M on spanning tree

For root P_r

1. Upon receiving no M

   Exit

   For each $P_i, 0 \leq i \leq n-1, i \neq r$

2. On receiving M from predecessor;
3. Send M to all its successor

   Exit

Algorithm 4.2: Broadcast on Existing Tree

is given, the time it takes to broadcast the message is depth of tree and message complexity is $n-1$. [20] This complexity is same for synchronous and asynchronous model.

4.3.3 Broadcast and Convergecast

Broadcast is the one way operation in which root propagate information, but this is not the only communication we required. In order to construct tree it is required to gather information from leaves and for this purpose we use convergecast with broadcast. Convergecast operation is initiated by leaves of the tree and it requires same time and message like broadcast. The general procedure is if a node i is a leaf then it starts the algorithm by sending its value suppose x_i to its parent. An intermediate non leaf node j, with k children, waits to receive messages containing $v_i 1 ... v_i k$ from all its children. And after recieving all messages from all children it computes the maximum or minimum among all messages and convergecast this maximum value to its parent. This method is also defined as collecting back acknowledgments to ensure that the broadcasting is accomplished correctly. If this operation is performed sequentially the time cost becomes high instead of that their is a technique defined by [2] denoted as, Pipelined broadcast and convergecast in which processors have different types of values and it is required to perform separate convergecast, the operation is pipelined in which processes execute parallel, each leave v initiate the k processes one by one. each intermediate non leave node computes each of the k partial maximum right after collecting the corresponding partial maximum from all its successors and convergecasts the maximum value to its father one by one.

4.3.4 Tree Construction using Broadcast

In this section we discuss the case in which we don't have any pre-constructed spanning tree for communication. In other words we want to produce a rooted spanning tree for broadcast. A simple flooding technique induces a spanning tree. Given a graph with set of vertices and edges, and a designated source P_s, flooding begins from P_s,then P_s broadcasts the message to all its adjacent nodes on all its outgoing edges. If P_u receives message for the first time, from some neighboring processor P_v, it floods message to all its outgoing edges leading to its neighbors except P_v. The message is sent once over each edge by processors, but there is possibility that message is sent twice on all edges except those in which message is received first time. If there are more then one message received by P_u than it chooses randomly. In the synchronous case this algorithm produces a breadth first tree(BFT) that takes $O(D)$ time, where D is the diameter of the network and $O(m)$ communication. In the asynchronous model the complexity is same but doesn't granted to be a BFT.[20]

Provide a graph G =(V, E) and a designated root p_r the procedure broadcast tree construction is defined in algorithm 4.3.

There are situation in which vertex of the network store data of different types and it is required to perform separate convergecast computation on each of the different data collection. Each such information is transmitted individually between different nodes of the tree. Even root have m distinct items to be shared with individual node. In a sequential procedure this process increases time complexity but in the distributed case this situation is handled by pipelining computation which reduces time as discussed before. Now we shall discuss other operations namely, downcast and upcast, which we required in a situation where we need to communicate different items on different vertices instead of a single message broadcast or convergecast.

4.4 Upcast and Downcast on tree

In the real network messages are not always of same type i.e. different vertices in a single branch of a tree possibly have distinct value to send to its parent. In the same way root may have different messages for its children in the tree. In that case we use downcast and upcast.

Downcast is the method for root node in such case when root has m different messages, each of which is destined to an individual node of the tree. A simple algorithm designed for downcast is, the root r randomly sends the messages for each of its children v. Its sends the message to subtree rooted at t sequentially on the edge (r_o, t). Each middle vertex in this process receives atleast one

input : G = (V, E) and a distinguish root r_o
output: A BFT rooted at r_o and each node having its parent
Initially variable $Parent_i$ = Nil, $Children_i$ = Nill and $Terminated_i$ = False and $other$ = Nil. A message M is placed in $outbuf_r$

if $p_i = p_r$ *and* $parent_i = Nil$ $Inbuf_i$ = *empty* **then**
  Send (M) to all outgoing edges (neighbors)
  $parent := p_i$
end
On receiving (M) from neighbors p_j:
if $parent$ = *Nil* **then**
  $parent = p_j$
  Send $(Parent)$ to p_j
  Send (M) to all neighbors except p_j
else
end
Send $(already)$ to p_j
On receiving $(parent)$ from neighbors p_j:
add P_j to $children$
if $children \cup other$ *contains all neighbors except* $parent$ **then**
  terminate
end
On receiving $(already)$ from neighbor p_j:
add p_j to $other$
if $children \cup other$ *contains all neighbors except* $parent$ **then**
  terminate
end

Algorithm 4.3: Tree construction with Broadcast

such message in each step and forwards this message towards the destination, until the node is itself the destination. The downcasting of m distinct message on a tree t requires $O(m+Depth(T))$ time in the worst case. Considering that there are no delays on message, the communication complexity of downcast operation is $t + Depth(T)$ in the worst case.

The upcast operation is used when leaves have m distinct items or some times more than one node has same data item. It is required to send all the data items to the root. This situation is quiet change from single convergecast as it sends combined acknowledgment,on the other hand in upcast each item is individually sent up to the root unless any vertex receives same message from its different children in that case it combined it and send single copy upwards. The upcasting of m distinct message to the root on a tree requires $\Omega(Depth(T))$ time.

There are few possible settings of items discussed in the literature, following are few of them;

4.4.1 Ranked Items

In this case if each single message is marked by its rank in the set than it can be easily upcast on the tree. Here ranking denote that items are given in the of pairs form (i, m_i) such that $m_i \leq m_{i+1}$ for $1 \leq i \leq m$. Upcasting of m ranked items requires time $Depth(T) + m$ on a tree t.

4.4.2 Ordered Items

The other case is when the items are not ranked by but they are still in some ordered set. We cannot tell about the position of particular item instead we can tell its property i.e. if its larger or smallest. For each vertex the procedure is in each round to upcast to father the minimum locally stored message which is yet not upcast by it, but vertex can see smallest items later on after upcasting previous smallest item. It also takes same amount of time as ranked items.

4.4.3 Unordered Items

In the case where items are absolutely non comparable except for there identity the unordered upcast is used in which vertex forwards randomly locally stored message. It requires $Depth(T)+m$ time to upcast m different unordered message to the root in the worst case.

An application of this operation is spreading distinct information among the vertices of the tree. For example some of the data items say, m are stored at the processors of the tree T. The goal is to disseminate information among all processors. i.e. each processor knows all the items. The cheapest way for accomplishing this task is to gather all the item at root and then root propagates each item one by one to all the processors in the network.

Now on we discuss number of distributed algorithms for tree constructions along with comparison of their sequential version. We consider breadth first tree construction, depth first tree, shortest path tree, minimum spanning tree. These trees provide structures for communicating efficiently.

Under the distributed setting we study some models for representation and assumptions on which the construction of network is based. First is LOCAL model, this model assumes the synchronous communication and allows any size of messages during communication and therefore it ignores the congestion and asynchrony factor. Other model is CONGEST that limitize the size of messages to avoid bottlenecks in communication. The standard message size is fixed to $O(logn)$ in this model. CONGEST model works on both synchronous and asynchronous model, and this is used in the case where the focus is on the problem of balancing time and message factor. Finally the ASYNC model that totally works on asynchronous principle and it allows any size of messages.

4.5 Breadth First Search

The first problem we consider is performing breadth first search to grow tree or predecessor graph. Breadth first search provides a structure which is convenient for efficient communication, broadcasting. We can use BFS trees for computation of global function, for leader election problem and for finding the diameter of the graph. In the synchronous case simple flooding algorithms produces BFT rooted at some source vertex, as defined previously.

The two basic algorithms for constructing the BFT for a given graph are Dijkstra and Bellman Ford. We are interested in distributed implementation of the breadth first search tree construction techniques, i.e. under the synchronous and asynchronous implementation. All of the algorithms assume that there is a designated source processor that begins the algorithm execution.

4.5.1 Layered BFS Algorithm

To the best of our research the idea of distributed BFS technique for level by level construction of the shortest path tree was presented by [24]. In which they discussed some algorithms for solving simple minimum hop problem and presented bounds for this problem. It also highlighted the trade-off between the time and communication complexity of the algorithm. Later this idea was improved to the distributed MST construction technique[7]. An important question in distributed computation is how to solve the problem, using a minimum number of messages. We requires communication efficient algorithm to construct network.

The dijkstra algorithm is the well known sequential algorithm for constructing BFS tree from a designated source node. It is also the generalization of BFS problem i.e. it computes the shortest path on weighted graph. The shortest path minimize the maximum cost of communication in the network. The weight on the edges may represent cost, delay. If all the edges are of equal weights then breadth first search tree is also a shortest path tree. The distributed version of this algorithm is defined as layered breadth first search algorithm which follows the idea of constructing tree layer by layer, from the root downwards. Each layer l consists of the nodes at depth l in the tree.

This procedure is a collection of phases, in the first phase, let r_o be the vertex initiate the process.r_o sends search message to all its adjacent neighbors and wait for acknowledgment. Vertices that receives the message in first phase send back positive acknowledgment at the end of the construction of layer 1, all the vertices at depth one knows their parent and parent knows its children. By assuming k phases have been completed in the same way, all the nodes at level k knows their children and parent nodes. To start new phase for $k+1$ layer the root r_o propagate new phase message along the channels of constructed spanning tree for the depth k nodes. Similarly when depth k

nodes found new phase message they spread message to all its neighbors but not to its parent. if the nodes receives the message for the first time it sends back positive reply and designate sender its parent. If a node receives a subsequent search message it replies negative. when a node at depth k got all its search replies it marks the node that have sent positive replies as its children.

To terminate the algorithm a variable is attached to the Convergecast Ack that shows that the new vertex is added to the tree or not. If the value of leaf vertex variable is 1 it means that it joins the tree otherwise its value will be 0 which shows that the next layer explored by the leaves is empty and the layered tree construction process terminates. [2, 3]

The problem of constructing distributed BFS tree have been widely studied in past and so far, due to the fact that this structure is the major building block for solving many other problems. We know that processors in the network are not aware of the topology of network each processor have knowledge of its adjacent processors. The source begins the algorithm and all other processors join in when they receive a message from one of their adjacent nodes. Eventually each processor knows its level and the shortest route (predecessor) tree is identified for each processor so that it knows which adjacent link are in the tree and which one of those adjacent tree links (inedges) are on the route to the root.

The basic coordinated distributed algorithm for constructing breadth first search tree requires $O(V^2)$ communication complexity and $O(VlogV)$ time complexity. An algorithm was proposed that takes $O(V^{1.6} + E)$ messages and $O(V^{1.6})$ time to construct BFS tree. [9]. The single source shortest route algorithm and BFS problem for a planar graph was also solved by [25] with a message complexity $O(n^{5/3})$.

The bellman ford algorithm in the sequential case solves the single source shortest route problem in case of negative edge costs. This algorithm works as follows; Each node keep record of distance from root i_o it knows so far. Initially $disti_o = 0$ and $dist_i = nil$ for $i \neq i_o$ and parent is undefined. In each phase each node send its $dist$ to all its outgoing neighbors. Each node update its distance by "$edgerelaxationrule$", in which it takes the minimum of its previous $dist$ value and all the values $dist_j + weight_{j,i}$,j is an incoming neighbor. The distance is updated if its changed and after $n - 1$ rounds, it outputs the shortest $dist$.[5, 3]

4.5.2 Update Based BFS construction

The Update based BFS technique is the distributed version of bellman and ford algorithm. The idea behind it is to run flooding to produce tree but this idea is not fine if communication is asynchronous. In asynchronous model it may happens that at some point when flooding progresses

some shorter paths are discovered due to the delay in message, therefore flooding is not proposed to be the solution for constructing BFT in asynchronous model. The basic idea behind update based solution is to store layer number. i.e. each node v records its variable layer no. $L(v)$ that indicates its distance from root node. So whenever any node find such shorter path later it resets its path along the one it belongs to and informs its neighbors about this action. It also update its level according to the correction. In synchronous model it takes $O(Diam(G))$ time and $O(|E|)$ messages to complete the construction of BFT with distributed bellman and ford algorithm. Furthermore for the asynchronous model the time required to perform this algorithm is $O(Diam(G))$ which is still same but the communication cost is $O(n|E|)$. Whenever a processor update its layer variable it informs this on all of its outgoing links therefore each processor shares at most $n.deg(u)$ messages until the termination of algorithm.

4.6 Shortest Route Tree

It is difficult to record the history of shortest route problem. As Compared with other combinatorial optimization problems like, minimum spanning tree, assignment and transportation the scientific research on shortest route problem begins comparatively late maybe because this problem is easy to solve. In the beginning heuristical, nonoptimal ideas have been suggested like, in 1956 Rosenfeld provided a heuristic for determining an optimal trucking path through a provided traffic congestion pattern. Route searching, in particular searching in a maze is related to the classical graph theory problem, Biggs, Lloyd and Wilson 1976 form a death first search technique.

In the beginning of 1950 the problem of finding path is studied in the situation of finding second shortest or alternative route in case of blocking of existing shortest route. It was first studied for the long distance calls in telephonic networks. Two simplest and well-known algorithms developed are, Dijkstra by Edsger Dijkstra in 1956, which is a graph search algorithm that solves the single source shortest route problem for a graph with positive edge path cost and bellman ford algorithm by Richard bellman and Lester Ford in 1958, that also provides the shortest route from single source to all other nodes in graph with negative path cost. It is not much faster than dijkstra's solution for the same problem but more adaptable as it can solves some of the negative costs on paths. Later running time of these algorithms is increased by using different data structures. Both of these techniques have already defined in previous section.

4.7 Distributed Depth First Search

The depth first search is another technique for exploring graph for finding a predecessor subgraph, it basically searches graph deeper and deeper and constructs a tree having the property of depth first. The procedure begins from root, reaching at a node v if v has neighbors that were not yet visited, then v next visit one of them. If there are no such vertex then v backtracks to the node from which it is visited first. This process takes $O(|E|)$ time in the worst case searching in the sequential manner. [5].

The distributed depth first search algorithm defined in [2] stated that message searching from vertex v to w in order to know that it has been visited earlier or not in any case has message and time of $\Theta(|E|)$. Its message complexity in worst case be always $\Omega(|E|)$ but time is improved by performing the exploration of all non-tree edges for free. To perform this it is required that a node knows about its neighbors being visited or not. Any node when first time visited, temporary stop the process, aware all its adjacent nodes that it has been visited, wait for acknowledgments form these adjacent nodes and then restart the search again. This schemes helps the algorithm to decide which vertex is next to visit. And the extra time cost required during each first visit to a node is $O(1)$ unit, and only tree edges are traversed, eventually the time it takes will be $O(n)$. [2, 3]

4.8 Minimum Spanning Tree

Another class of tree is minimum spanning tree (MST), this is said to be minimum because its total cost is minimum among all other spanning trees of graph. Weight on the edge is a given parameter for evaluating and computing spanning tree, here we discuss two best algorithms for constructing MST and also a distribute algorithm for the same problem. In all cases it is assumed that costs on edges are unique. There are number of applications where MST algorithms are helpful, such as solving the leader election problem in general graph, for broadcast communication. All the algorithms for constructing MST,s weather they are sequential or distributed work on the same pattern i.e. initiating by creating n nodes component and repeatedly merging components with the minimum weight outgoing edge (MWOE) until single component left. [2, 6]

4.8.1 History of MST: Prim's and Kruskal's Algorithm

The first algorithm for finding MST was developed by Boruvka in the year 1926, the idea was finding the safe edge joining vertices until a tree spanning all vertices complete. After that each variation of finding MST was based on the way by which that safe edge was selected. The second

technique was known as Prim,s algorithm which was basically created by Jarnik in year 1930, and later improved by Prim in 1957 and then further enhanced by Dijkstra in 1959.

The Prim's techniques works as follows; Starting at any node v in the graph and observing the least outgoing edge of v to any neighbor u. It connects v to u with this minimum weight edge. After that algorithms finds the node w which is either the neighbor of v or u with the property of minimum weight edge between them. It adds w to the tree and grows the tree. In this process each node with such property is added to tree and considering that no cycle is there in the process it eventually connects all the vertices hence MST is produced by algorithm. If there is a case when two or more edges have same cost then any one can be chosen randomly. The time complexity for solving this problem depends on the particular data structure used.

The third famous algorithm is known as Kruskal's algorithm. For MST construction it takes n nodes, one by one grows the tree by adding the minimum cost edge. If n–1 edges have been added, the MST is set. As in prim's algorithm, If there is a case when two or more edges have same cost then any one can be chosen randomly avoiding cycles in the graph. [5]

4.8.2 Distributed MST "Gallager Humblet and Spira" Algorithm

The most famous and distinguished achievement in the history was the algorithm proposed by [7] for finding MST in distributed network. The basic idea of work is to join small components into big components with minimum cost outgoing edge. A component which is also called fragment is a subtree of the large MST. Each node, in the start is a fragment and then algorithm repeatedly merges fragments until there is only one fragment left which is MST. It takes $O(E + n \log n)$ messages and $O(n \log n)$ time to construct MST through this technique.

The author stated three possible acts of nodes i.e. sleepy, find and found position. In the start each processor state is sleepy and it is either alive by initiating the algorithm or by receiving message from any other processor. When the processor is in find state it finds the MWOE to join the other fragment. Because each node is initially a fragment so it is also the root of its fragment. Fragment joining rules follow some levels.

A fragment with a single node has the level $L = 0$. Suppose two fragments F at level L and F' at level L';

1. If $L < L'$, then fragment F is immediately absorbed as part of fragment F. The expanded fragment is at level L'.
2. Else if $L = L'$ and fragment F and F' have the same $MWOE$ than the fragments combine

immediately into a new fragment at level $L + 1$.

3. Else fragment F waits until fragment F' reaches a high enough level for combination.

In the light of these rules the joining channel is called the core of the new component. Two nodes adjacent to the core share messages on the core branch itself, permitting each of these nodes to know both the cost of the MWOE and the location of the core on which this edge exist. The time required by this algorithm is $O(E + N \log_2 N)$. This algorithm is formally written in preliminary chapter.

The problem of searching MST in a distributed computing has been studied since 1977. A significant and unique work was presented by [7] discussed above. Later on many other researcher improved the bounds which sightly changed the results like the algorithm presented by, [28, 29] has time complexity $\Theta(V \log^* V)$, while the message complexity was still the same. Another scheme proposed by [8] is the improvement of [7], in which each tree will hang itself on link leading to the neighboring tree of maximum level instead of hanging itself on MWOE so in the case tree waits for a long period it is compensated accordingly. This is the main idea behind the counting stage of their technique. This algorithm has a lot similarity with GHS but the main difference is that level increases are initiated by more the one nodes, not only by the source. Their proposed technique has two stages; the first process is similar to GHS and terminates when all trees reaches the size of $\Omega(n/\log n)$. The second phase introduces the new ides in which algorithm updates the level in a very convenient manner, which stops small trees to merge with big trees and increases the speed of construction. The time complexity of this technique is $O(V)$ and it takes $O(E + V \log V)$ communication.

4.9 Maximal Independent Set

In this section we discuss another graph algorithmic problem that is, Maximal independent set construction. This is also listed as the efficient symmetric breaking technique and basic building block for several other problems. The nodes in a given set are said to be independent if they no single node having its neighbor in that set. In other words no two adjacent nodes are in the independent set. Maximal is the property of any object which shows that it is not further increased to form a larger object by adding any other object. An Efficient MIS construction algorithm can also used to decomposing the network in to clusters of low diameter. The simple sequential efficient method for finding MIS is as follows;

Procedure Finding MIS

Given a graph with (V, E), we consider set U, initially contains all the vertices of the graph and

set M that is maximal independent set, initially empty. While U is nonempty we take vertex v from U and add it in M, and remove vertex v and all its adjacent vertices from set U. M forms an independent set after this procedure terminates it has linear $O(|E|)$ time complexity. Several distributed implementation of this procedure has been presented like the algorithm presented by [2] in which the output of the MIS computation is checked by a variable, and vertices change its value according to the situation. The distributed implementation is based on the depth first tour of the network in which control is carried with the message passing, other procedure is same as sequential algorithm. In the distributed case this technique takes $O(|E|)$ messages and $O(v)$ time to complete.

The problem of partitioning network into clusters is same as finding maximal independent set. The set of nodes in the independent set is also called dominating set. The concept of weighted clustering technique [41], depends on some attribute of node for example lowest or highest id or node connectivity i.e. the degree of node in the graph. These are two traditionally being used attributed for selecting node for the role of clusterhead. A communication round is required to find the corresponding weight of the mobile nodes and this communication delays actual tasks such as routing, scheduling etc. We need such clustering schemes that reduces the communication round and results fast clustering formation.

A technique given by [42] uses both of these parameters for selecting clusterhead for clustering in ad hoc wireless network. A node with the lowest id in its neighborhood broadcasts clusterhead message to its neighbors but before that, a node with the maximum degree will create a cluster around it. In case degrees are same then lowest id will be checked. An entropy based weighted clustering [43] improved the overhead of WCA by calculating mobility, consumed power and entropy of the node for clusterhead election. An improved version of simple MWIS [39] technique was presented by [44] known as GMWISA, which is based on adjacent and non- adjacent vertices set. It calculates the weight of each node based on AV and NAV list and assigns the role to each node accordingly. The GMWISA computed independent set faster then conventional MWIS algorithm.

Recently a modified weighted clustering algorithm [45] was proposed with some unique definition of attributes of nodes. The author efficiently differentiate between relative and absolute information of nodes that are consider during cluster formation. Absolute attributes of any node are mobility, energy consumed and entropy of node whereas relative attributes are connectivity of node, sum of neighbors etc. They used absolute attributes information for cluster formation because such information requires measuring node parameter locally rather than neighbors topology information. For absolute weight calculation they used entropy based clustering [43] in which mobility, power consumed and entropy of mobile node is calculated these attributed do not requires neighborhood

calculation. The minimum weight among them is elected as a clusterhead. They improved cluster formation process by using only absolute attributes information.

Another well establish technique for symmetry breaking is coloring technique, there is a lot of common between these two methods, vertex coloring methods uses set of some color to color each vertex of the graph such that no two neighboring nodes have same color. A chromatic number is defined as the minimum number of color that can be used to color each vertex of the graph. The problem is to minimize the number of color. The coloring problem can be reduce to the MIS. In which we know that vertex v that joined set M cannot be adjacent to those vertex that have already joined M and even not to those vertex that are in progress of joining M. If this happens then the procedure is not growing the MIS. for more detail we suggest [2, 3]

In next section we discuss the pure locality sensitive representations of network in order to integrate it with distributed network algorithm. The main source of this discussion is [2]. They presented number of ways to represent network graph, namely cluster based representation, cover and partition. The main idea of this work is to break the system in to small unit clusters and then create a cover by different methods and properties. The technique for constructing average and maximum sparse cover is discussed and also sparse partitioning was introduced. All the above defined methods are based on the core idea of coarsening. i.e. merging clusters in to other by some property.

Initially we formally define these terms as they are defined in the original work; A cluster is an induced subgraph which is connected. Given a set of nodes $S \subseteq V$, let $G(S)$ denoted a subgraph induced by S in G, namely $G(S) = (S, E')$, where E' consist of all the edges of G whose endpoints both belongs to S.

A collection of clusters covering all the nodes of graph is said to be a, cover denoted as $\mathcal{S}$ such that $\mathcal{S} = \{S_1, ...S_m\}$. On the other hand a partition of G is defined as collection of disjoint clusters such that $S \cap S' = \emptyset$ for every $S, S' \in \mathcal{S}$. For clustering and partitioning the two important evaluation criteria is cluster locality level or more general the size of cluster, and the other is sparsity. These two parameters are naturally conflicting and the basic problem is to balance these two factors. For defining the partition sparsity, each cluster is considered as a vertex and defined as cluster graph or super graph, formally the cluster graph of $\mathcal{S}$ is defined as;

$$\widetilde{G}(\mathcal{S}) = (\mathcal{S}, \widetilde{E})$$

where

$\widetilde{E} = \{(S, S') | S, S' \in \mathcal{S}$, and G contains an edge (u, v) for $u \in S$ and $v \in S'$. The edges of $\widetilde{E}$ are called intercluster edges.

Minimizing the number of intercluster edges is maintaining sparsity. For understanding sparsity of cover and partition they defined several other properties like vertex degree, cluster degree, average and maximum case, Vertex and cluster neighborhood. All the definitions based on the concept of neighborhood. They defined $\mathcal{P}$- neighborhood of vertex as the set of nodes at distance $\mathcal{P}$ or less from it in graph. Neighborhood is also denoted as $\Gamma^{v}_{\mathcal{P}}$. i.e. vertex neighborhood at distance $\mathcal{P}$. There are different sparsity measures for cover and partition. We define each of them according to our need.

4.10 Partition: Definitions for measuring Locality - Sparsity

The cover and partition are the two basic representation of network both have different sparsity measures. We focus on partition of network i.e. clustered construction, sparse partition of network. The goal of our study is to balance the locality/radius and sparsity trade-off in the resulting structure. Now we shall focus on the measure of Locality and sparsity of a partition.

The partition of a graph is the collection of non intersecting clusters. For measuring the cluster locality one of the important concept is neighborhood or more specifically vertex neighborhood clusters. The $\mathcal{P}$ neighborhood concept was introduces in this term i.e. a vertex neighbor at distance $\mathcal{P}$. For the cover this neighborhood concept is extended.

Given a subset of vertices W of V, the $\mathcal{P}$ neighborhood cover of W is the set of $\mathcal{P}$ neighborhood of the vertices of W denoted;

$$\Gamma_{\mathcal{P}}(W) = \{\Gamma_{\mathcal{P}}(v) | v \in W\}$$

For the partition sparsity measure - adjacency, clusters are contracted in to super nodes.. Each cluster is a node of a cluster graph $\widetilde{G}(\mathcal{S}) = (\mathcal{S}, \widetilde{E})$ and the edges of the cluster graph are the set of intercluster edges.

Cluster Neighborhood

Given partition $\mathcal{S}$, cluster $S \in \mathcal{S}$ and integer $l \geq 0$, cluster neighborhood of S is equal to neighbors of S in cluster graph $\widetilde{G}(\mathcal{S})$ represented by;

$$\Gamma^{c}_{l}(S, G) = \Gamma_{l}(\mathcal{S}, \widetilde{G}(\mathcal{S}))$$

Similarly a vertex neighborhood of any cluster S in a given partition $\mathcal{S}$ is the combination of the $\mathcal{P}$ neighborhoods of the nodes in S.

A cluster degree is the number of its outgoing edges. Now we define the average cluster degree of partition $\mathcal{S}$ which also tells the number of inter-cluster edges.

Counting Intercluster edges

Let us now discuss the avergae cluster degree of partition $\mathcal{S}$, which is also used to find the number of intercluter edges. The average cluster degree of partition $\mathcal{S}$ is;

$$Av\Delta(\mathcal{S}) = \Sigma_{S\in\mathcal{S}}|\Gamma^C(S)|/n$$

which also produces the number of intercluster edge of S. The goal of any partition algorithm is to produce the partition with specifed radius and few no. of intercluster edges.

4.10.1 Basic Partition

We start with the simplest method for construction of sparse partition of graph by [2]. Given an simple non weighted graph $G = (V, E)$ and a parameter $k \geq 1$. The algorithm produces a partition $\mathcal{S}$ with cluster radius at most k and with the few intercluster edges. The algorithm $BASIC - PART(G, k)$ is as follows;

input : A Graph G = (V, E), $k \geq 1$
output: Partition $\mathcal{S}$ of G
Set $\mathcal{S} \leftarrow \emptyset$
while $V \neq \emptyset$ **do**
  Randomly select a vertex $v \in V$
  Set $S \leftarrow \{V\}$
  while $|\Gamma^v(S)| > n^{1/k}|S|$ **do**
    Set $S \leftarrow \Gamma(S)$
    Set $\mathcal{S} \leftarrow \mathcal{S} \cup \{S\}$ **and** $V \leftarrow V - S$
  end
  Output(S)
end

Algorithm 4.4: $BASIC - PART(G, k)$

The algorithm starts by taking a vertex v from network and grow the cluster around it by putting layers one by one. The vertex added in cluster S are at the same time removed from V. Layer merging process carried out repeatably till the next round increases the number of modes merged into S by a factor of less than $n^{1/k}$. The procedure adds the resulting cluster S to the output set $\mathcal{S}$, and then new round begins and this process ends when V is finish. In the above partition radius of a partition is less then or equal to $k - 1$. And the cluster graph $\widetilde{G}\mathcal{S}$ contains maximum $n^{1+1/k}$ intercluster edges.

4.10.2 Sparse Cover and Partition

Coarsening is the technique of subsuming or merging clusters together. The technique for constructing sparse cover relies on the idea of coarsening clusters. Formally;

If we have two covers $\mathcal{S}$ and $\mathcal{T}$ it is said that $\mathcal{T}$ merges $\mathcal{S}$ if the clusters of $\mathcal{S}$ are completely subsumed in $\mathcal{T}$. This concept is use for sparse cover construction. The procedure of cover construction by coarsening is defined next. The theme of this algorithm is same for several other vairants of this problem. The general procedure is as follows;

$\mathcal{T} \leftarrow \emptyset$
while $\mathcal{S} \neq \emptyset$ **do**
  Select randomly a cluster $S_0 \in \mathcal{S}$
  $Z \leftarrow S_o$
  repeat
  $Y \leftarrow Z$
  $\mathcal{Z} \leftarrow \{S | S \in \mathcal{S}, S \cap Y \neq \emptyset\}$
  $Z \leftarrow \cup_{S \in \mathcal{Z}} S.$
  until $|Z| \leq n^{1/k}|Y|.$
  $\mathcal{S} \leftarrow \mathcal{S} - \mathcal{Z}.$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{Z\}$
end
Output($\mathcal{T}$)

Algorithm 4.5: $AV - Cover(G, k)$

Each round builds one output cluster $Z \in \mathcal{T}$ by merging in it some clusters of $\mathcal{S}$. The round starts by selecting randomly a cluster S_o in $\mathcal{S}$ and assigned it a starting cluster point. The chosen cluster is after that back to back merged with intersecting clusters from $\mathcal{S}$.

At each point the core is viewed as the internal kernel cluster Y of the output cluster Z. The merging is back to back done till the required parameter found. After that algorithm includes the final cluster Z to the output set $\mathcal{T}$, the new round starts until the $\mathcal{S}$ finishes. Finally algorithm outputs the set $\mathcal{T}$. In this procedure $\mathcal{T}$ coarsens $\mathcal{S}$ and the radius of $\mathcal{T}$ is less than or equal to $(2k+1)Rad(\mathcal{S})$ and the number of intercluster edges is less than or equal to $n^{1/k}$.

On this base they also defined the procedure for partial partitioning, and also maximum degree cover construction algorithm and even for sparse partition construction.

The base paper that produces the near linear cost sequential and distributed algorithm for constructing sparse neighborhood cover was [30] with the time complexity $O(E \log n + n \log^2 n)$.

4.11 Network Decomposition

The concept of network decomposition was found first time in [31]. Network decomposition can be used for designing deterministic algorithm for problems like MIS construction, coloring with low overheads depending on the parameter of decomposition. The concept of the first proposed solution for decomposing a graph is to divide graph in to small connected components that have low diameter in a way that if each component is considered a single node can be colored with a less number of colors. They also presented such type of decomposition for finding Maximal independent set that runs in $O(n^e)$ time for e = $O(\sqrt{\log \log n / \log n})$. Later they proposed deterministic sub linear time algorithms for distributed network decomposition and for growing a sparse cover of a network.[32] They successfully make use of coloring technique for decomposition and improved preprocessing time of distributed algorithms like all pair shortest path, balancing load in network, broadcast and bandwidth management.

Moran [26] proposed a solution for constructing efficient network decomposition and using this decomposition they present variants of the synchronizers, Particularly in an asynchronous network they perform BFS without any preprocessing done. Its communication and time complexities is $O(K|V|D + |E| + |V|log|V|)$ and $O(Dlogk|V| + |V|)$ respectively, for $K \geq 2$ provided a parameter. Additionally, they proposed an improved version of cluster coarsening algorithm for merging clusters efficiently.

An easy distributed implementation of algorithm maximum partition by [1] can also be used to produce the much faster preprocessing algorithm for constructing synchronizer γ by [33]. The goal of designing synchronizers is to make it possible for synchronous algorithm to run in an asynchronous network. They discussed two synchronizers and presented third synchronizer which is combination of the previous. The synchronizer γ starts when network is partitioned into clusters, just like we discussed the node partitioning method. For each cluster they choose preferred edge as a intercluster edge. Each cluster has a leader that coordinates operations via intracluster tree. The partition algorithm they defined is as follows:

The heart of any partition algorithm is cluster construction by the procedure defined by [33] a node which is chosen as a cluster head starts execution of the algorithm (BFS). Each layer of BFS merges with the the existing cluster until the number of nodes in the certain layer is less than $k-1$ times the sum of the nodes in all the previous layers. The procedure stops at that point and then the search for new leader begins.

The distributed implementation of above idea is presented in [33]. The cluster construction follows distributed breadth first search idea for layer by layer construction of tree. The decision of any node

whether it is going to join layer l of the cluster or not depends on the number of nodes in that layer. The leader broadcast next layer message along the tree on existing layers, when message reaches to the leave nodes i in the last layer it sends message to its neighbors about its position in the cluster. On receiving such message neighbor j which is still alone, joins cluster layer and consider i its parent, j sending an Ack bit to i carrying 1 if it chooses i its parent or 0 if i was not chosen as its parent. For counting the no. of new nodes and to assure that all nodes are counted again convergecast is used. Each node sends the no. of its children in new layer to its parent inserting it into the count message. finally the leader knows the no. of nodes at new layer. If the no. of nodes in new layer are $k-1$ times high than existing nodes of the tree than the next pulse starts and nodes of new layer are informed that they now part of existing cluster. In the the case leader sends reject message to that layer. The cluster construction stops and when it stops each node knows itself and its neighbor (joined or not to any subgraph).

The new cluster leader is selected from the rejected layer of cluster C, if it is not empty. If that layer is empty the control is given back to the cluster from which C was found and the above defined procedure is repeated on that location just like depth first search. New leader election procedure also starts at leader, that broadcasts TEST message on tree edges. A node in the least layer checks its neighbors and selects the one whose id is lowest among all all other neighbors. It sends its selected choice to its predecessor in tree. In the same manner each middle vertex upcasts its local selection of node to its predecessor eventually at root node it ends up with the lowest id node in the rejected layer. If that layer is empty it ends up with no option. Then root broadcasts the new leader message to that particular node having minimum id. When that node receives New leader notification it begins growing its own cluster by the procedure defined above. This node also remembers the cluster from which it received the notification that it you are new leader. Whenever the layer is found empty the process backtracks to its parent cluster with a Retreat message. This process is repeated upto the very first cluster than the whole graph search terminates and no new leader found.

After this the preferred link selection procedure starts, weights are associated with each link i, j and the minimum weight edge between two neighbor clusters are selected. This election procedure is handled by each cluster separately, this assures that two clusters selected the same min weight edge between them. When the search for new procedure backtracks we observe that all the nodes in neighboring have already been join to clusters. Again the leader broadcasts $Election$ message along the tree. This process is handled by convergecast process. Each node sends election list to its parent, which is created by it, after gathering lists from its children in the tree. This list specifies for each cluster neighboring one of the above nodes, the minimal cost link going to it. Leaves of the cluster tree convergecast their local list to their parent, a middle node combine these lists with

its own list. When these lists are gathered the middle vertex upcast its own list to its father, this process ends at leader when the leader gets lists from all its children merged with its own list. After that the leader propagates the final list along its tree.

4.12 Routing in a distributed network

One of the fundamental problem in distributed network is message routing. It is basically selecting path in a network through which two devices can communicate, but this selection is not so easy, the mechanism for supporting this task is called routing order. This order consists of message forwarding rules, tables containing data. In this section we discuss structure of routing schemes and geometric routing problem in detail. The source of this literature is [2, 27]. The natural desire is to route message on minimum possible shortest path. Our one basic goal in large networks is to construct such type of routing scheme that takes minimum memory and produces efficient route. This problems is studied extensively in the past considering the efficiency, memory trade off which is much similar to the radius sparsity tradeoff for constructing network partition and cover.

4.12.1 Routing Schemes

The routing schemes consist of data structure and data forwarding algorithm. The data structure is composed of three functions, one for each vertex v. An initial header function, I_v : $Labels \rightarrow Headers$. Another header function H_v : $Headers \rightarrow Headers$ and a port function F_v : $Headers \rightarrow [1, deg_G(v)]$. The forwarding algorithm defines for each node to forward message under some rules. [2] The shortest path routing is the main focus of interest, because we always want the route with the minimal weights. Construction of shortest path routing order required couple of tasks; first is constructing the shortest route and then after that saving processed data on different locations so that nodes can use this information to exchange messages. For storing the route, full table routing scheme is used to save a complete routing table at each node in the network. The performance of route of any routing order is checked by dilation factor. It is the percentage between the communication charges of routing a message from the starter r to the destination v using the scheme and the minimal charge of passing the same message on the same nodes. The maximum such percentage over all possible starter-destination pairs is the dilation. The other important factor is memory requirement of the individual node for storing routing scheme function. For any routing scheme these are the major two quality measures.

4.12.2 Distributed Geometric Routing

Geographic routing often called location or position based routing is one of the types of routing in which a node u sends message to node t on the basis of the information of its geographic location. This geographic position is determined by the nodes before they communicate with each other.

Earlier approaches of geometric routing were based on greedy forwarding in which the packet is forward to the node which is nearest to the destination. Compared to other techniques it has low overheads and it performs well when the size of the network is large. However, greedy routing may lose to deliver a message, because the message may reach a node whose neighbors are all farther away from the destination. According to the literature survey, geometric routing was proposed by [34]. Most of the earlier geographic routing algorithms are based on greedy forwarding or flooding strategies the problem with these techniques are they don't guarantee message delivery and often create network congestion problem due to sending multiple copies of the message.

A technique called planar graph routing or face routing was developed to overcome the problem of greedy routing. Through the literature survey, the first algorithm that ensures message delivery without using flooding was presented by [35] The algorithm initiates from source s, suppose that there is a F is the face intersected by line segment that meets s and the destination z $\bar{sz}$. The algorithm traverses the edges of F and keep tracks the intersecting area w on line $\bar{sz}$ along the edge which is closest to z. The algorithm after exploring all the links visit back to w. In face exploring process if the destination is found then algorithm terminates. If the destination is not found then w divides the line $\bar{sz}$ into two segments in which the line zw is still unexplored part of line $\bar{sz}$. The face is then updated to be the face that is incident to the w that is intersected by the line wzin the area of w and begins the rest again. The compass routing was also proposed in the same work. The basic point they proved that the compass routing also works for delaunay triangulation. In this technique if a message wishes to reach the intended node say t from any source r. Then the algorithm will broadcasts this message to the neighbor s of r such that the slope of the line segment joining s to r is nearest to the slope of the line segment joining r to t. If the delaunay triangulation graph is used then compass routing will possible not fails to deliver message to the destination.

Another efficient routing algorithm was presented by [36]. They presented the first distributed algorithm for routing which omits duplication of messages and memory at nodes and results better performance.They used a local planar subgraph construction based on the gabriel graph. After this another technique with the name "adaptive face routing" was developed by [37] along with the initial worst case analysis of geographical routing techniques. The algorithm presented by [37] was the first competitive algorithm with respect to the shortest route between initiator ant intended

node. It basically enhanced the face routing algorithm by the idea of bounding ellipse restricting the searchable area thus AFR is asymptotically optimal.

A distributed geometric routing algorithm proposed by [27] is based on the idea of locally constructing delaunay triangulation to construct a planar graph. In both cases two local triangles may intersects or the local triangle or Gabriel edges may intersects. Neither this algorithm nor the previous version of this work can guaranteed planer graph. The key feature that differentiates this algorithm from the previous strategy is that instead of removing the triangle the proposed algorithm removes all edges in the neighborhood in case when a node u found such node in local triangle in which there exist an intersection between the local triangle and the Gabriel edge. After discarding those edges each node will construct localized delaunay triangulation in the neighborhood. Eventually, node u has Gabriel edge and all triangles i.e. edges belong to non- intersecting triangle.

The basic idea is taken from the technique in which Gabriel graph is used along unit disk graph and proved that MST is a subgraph of the unit graph. In the same way this technique is applied in proposed algorithm because the minimum spanning tree is the subset of delaunay triangulation so algorithm finds the intersection between the unit disk graph and the local delaunay triangulation to present that the resulting subgraph is connected, if the intersection of UDG and the graph produced by the algorithm is connected. This idea is proved in the paper. The algorithm is divided into two phases, phase 1 produces the connected planar graph form the original graph and the other phase performs routing on that graph.The idea was based on finding the Gabriel edges and constructing a delaunay triangle to get the planar graph. In the first step each nodes gathers local information. By using this knowledge each node computes local delaunay triangulation with the property that edges of the triangles are not higher than one unit. In the next step non Gabriel links are removed through the communication between nodes. Eventually the planar graph is produced by the algorithm. On this planar graph face routing algorithm is executed. The message complexity of this algorithm is $O(d \log d)$ bits, where d denotes degree of a node. They sets the lower bounds for this problem.

5 Problem Formulation

5.1 Introduction

In this section we formally define our thesis problem by modeling it on network. Finding an efficient and convenient structure for communication is a well studied problem in past and so far. The construction of sparse network is inspired by [1, 2] in which they presented variety of techniques for constructing efficient network, such as sparse cover, sparse partition,and also network decomposition techniques. The basic problem around all these constructions is controlling locality and sparsity. Specifically if we consider clustered representation of network the two major criteria of measuring its efficiency is its size, which is captured by its radius and sparsity i.e. interaction between clusters. These terms are also defined as, latency and load. Latency is defined as the total time it takes for a message to cross the network from a node that propagated that packet to the farthest intended/destination node. Delay could be considered as a synonym of latency. Load refers to the amount of data (traffic) being carried by the network. Our problem deals with balancing load and latency of the network and providing the efficient structure for communication since these two parameters are naturally conflicting i.e. improving on of them results degrading the other. The sparsity is captured by decreasing number of degrees, it can be vertex degree or cluster degree. These two measures are strongly related to each other while we discuss the complexity of algorithm just like if the size of cluster is small it results low time complexity because computation is perform inside cluster on the spanning tree on the other hand if the degrees of vertices are low than they require low memory because vertices store information only for those clusters which they belong to. The message complexity depends on both of these parameters.

5.2 The Network Model

The communication model consists of a weighted graph $G(V, E, p, w)$ where V and E are the set of vertices and edges respectively and $p : V \rightarrow R^+$ is the function assigning a positive real valued transmitting power to each vertex $v \in V$. Each communication link $e = (u, v)$ has an associated non negative real valued delay; $w(e) = w(u, v)$ in the network. Furthermore each processor of the network G has its unique identifier assigned to it [lemma:Impossibility Result].

5.3 Thesis Problem

The problem of constructing efficient network in terms of balancing network load and latency is formalized as follows;

Formally: Given wireless network nodes with their unique IDs and incident edges (communication links) on a plane. We assume that each vertex of a graph v has an associated unique non negative real valued transmitting power; $p : V \rightarrow R+$ in the network. Our thesis problem is to build a convenient network structure (sparse network) for efficient broadcast communication in a localized manner on a synchronous distributed model of computation such that network latency and load are optimal (balanced).

This is one of the classical network design problems that demands efficient solution. We are dealing this problem on distributed computing in a localize manner. In preliminary section we gather enough definitions or concepts to simplify our problem.

6 A Proposed Distributed Algorithm for Broadcasting

6.1 Introduction

This chapter presents an efficient distributed algorithm to build a convenient network structure (sparse network) for efficient broadcast communication in a localized manner on a synchronous distributed model of computation. The divide and conquer technique is traditional being used for many distributed problem. Our straight forward approach is partition the network in to small set of nodes and than achieve desired level of interaction between these subsets. This type of partitioning is usually referred as cluster based hierarchical representation of network. This idea is due to the great work by [1]. And the distributed techniques used to solve the problem are based on [2, 3, 7, 20, 39, 41].

6.2 Decomposition of Network into Clusters

In this section we present a solution for decomposing a network into clusters of low radius. The basic purpose of our construction is to balance load and latency of network by partitioning into suitable clusters. We reduced the problem of partitioning the nodes of the network in to the problem of searching maximal weighted independent set of nodes, inspired by idea presented by [39]. We use the nodes transmitting powers which is represented as a unique positive integer on nodes, to determine the eligibility of a node for the role of clusterhead. In a distributed network each node has a unique identification, which is known to a node of same class often known as neighbors. This idea and technique is taken from [39].

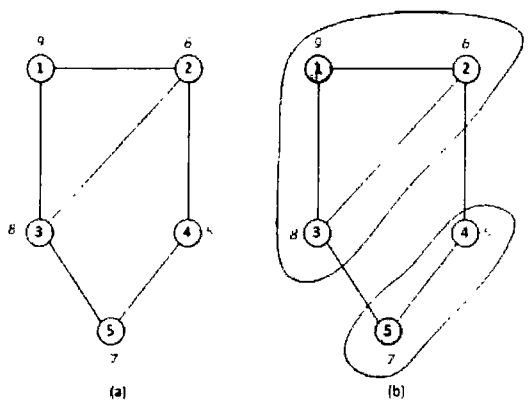


Figure 4: (a) A wireless network G with vertices v and their transmitting power(b) A partition of G into clusters, grey colored nodes belongs to the set of clusterhead.

6.2.1 Algorithm description

This algorithm basically provides the solution for determining the set of clusterheads of a suitable clustering for wireless networks. At the end of the execution the network results partitioned in to clusters, each of which has one independent node which is clusterhead and some ordinary nodes which belongs to some cluster. The algorithm is distributed i.e. it executes at each processor and each processor determines its state based on its own information as well as information of the nodes of same class. For this purpose we design set of messages which are used in the process of clustering.

6.2.2 Specific Messages

1. Broadcast message denoted as $bcast(p)$, initial message through which each node sends its own transmitting power p to its neighbor.
2. Clusterhead message denoted as $Ch(v)$ A message send by node v declaring it self a clusterhead.
3. Connection message denoted as $Cnct(u, v)$ send by a node u to aware all nodes about its connection with cluster whose clusterhead is v.

The algorithm is synchronous and solely based on messages, i.e. a particular procedure will run at a node depending on receiving the corresponding message. The idea of using this algorithm and technique is due to[3, 2, 39]. We consider transmitting power of nodes as a parameter for selecting clusterhead. We avoid choosing other relative attributes such as id or connectivity because these attributes delay the cluster formation which delays the actual operations. The transmitting power is calculated locally and the maximal transmitting power node is selected as a role of clusterhead. Transmitting power based cluster formation technique is simple and long lasting because every time the node with the highest power is elected therefore the network stability becomes high.

Each node of the network starts the execution at the same time by running the procedure as follows: Each node broadcast its transmitting power to its neighbor but only those nodes that have bigger transmitting power among all their neighbor will propagate $Ch(v)$ message to its adjacent nodes. There always exist atleast a node v that sends this message because of unique transmitting powers. Rest of the nodes waits for receiving message.

Upon getting Clusterhead notification from one of neighbor u, every node v checks if it has got from all its neighbors z such that $p(z) > p(u)$, a Cnct(z, x) message , where x is any other cluster. A Cnct (z, x) message means that all those nodes have already joined some cluster x. In this

situation v will not receive such Clusterhead notification from these neighbors and u is the node with biggest transmitting power in v,s locality that has sent a Clusterhead message. v decides to join u and quits execution. But if there is still at least a node z, $p(z) > p(u)$, that has not sent a message yet, node v just keep track of this in the variable Ch(u) that u sent a Clusterhead message and keep waiting from a message from z.

On getting Cnct(u, t) message, node v determine the kind of messages received so far from its locality z such that $p(z) > p(v)$. If v has received a connect message from all its neighbors with bigger transmitting power, this means all these nodes have already joined other clusters. In that situation v will be a clusterhead because now v has the bigger transmitting power among all its neighbors. But in case v has got at least a clusterhead message from any neighbor z with $p(z) > p(v)$ then it checks if all the neighbors u with transmitting power greater than $p(z)$ have sent a Cnct message. In this situation v can join the cluster of the biggest z that have sent a clusterhead message and terminates the algorithm execution [39].

6.3 Proposed Clustering Algorithm Steps

In this section the algorithm cluster formation is formally describe as an I/O automaton.

Algorithm: Graph Clustering Automaton

For solving the problem, as an input we have a network G with set of vertices V and associated links E. Each node v of the network is assigned a unique positive real valued weight representing transmitting power and unique IDs. We describe the algorithm as an I/O automaton i.e. Set of input, output, set of states a moving function and tasks.

Input :

$bcast(p)$ where p is the transmitting power of v

receive $Ch(v)$

receive $Cnct(u,t)$, $t \in nbrs(u)$ and t is clusterhead

Output:

send $Ch(v)$

send $Cnct(u,t) t \in nbrs(u)$

States

- v, a generic node encodes both nodes ID and transmitting power, $p(v)$
- A boolean variable $Ch(.)$ initially false. For a node v, $Ch(v)$ will be true if either it broadcasts $Ch(v)$ message or either it gets $Ch(u), u \in nbrs(v)$, message.
- A boolean variable $Cnct(.,.)$, initially false. For a node u, $Cnct(u,t), u \in nbrs(v), t \in V$ is marked to true by v when it receive the $Cnct(u,t)$ message from u.
- An $Exit$ state in which node stops further execution.

Moving Function

$bcast(p)_{v,u}, u \in nbrs(v)$

Effect:

if *for each* $u \in nbrs(v), p(v) > p(u)$ **then**
| send $Ch(v)$;
end
$Ch(v)$ = true
Exit

receive $Ch(u)$
Precondition:
if $Cnct(z,x) =$ *true,* $z \in nbrs(v), p(z) > p(v)$ *and* $x \in V$ **then**
  Effect:
  $Ch(u)$ = true
  send $Cnct(v,u)$
end
EXIT

receive $Cnct(u,t)$
$Cnct(u,t) =$ true;
Precondition
if $Cnct(z,x) =$ *true,* $z \in nbrs(v) : p(z) > p(v)$ *and* $x \in V$ **then**
  Effect
  send $Ch(v)$
  Ch(v) = true;
  Exit
else
  Precondition
  if $Ch(z) =$ *true,* $z \in nbrs(v)$ *and* $Cnct(u,x) =$ *true,* $u \in nbrs(v), p(u) > p(z)$ *and* $x \in V$ **then**
    Effect
    send $Cnct(v, maxp_z\{z : Ch(z)\})$;
  end
end
EXIT

Tasks

For every $u \in nbrs(v) bcast(p)_{v,u}$

send $Ch(u)$

send $Cnct(u,t)$

Algorithm 6.1: Automaton: Graph Clustering

6.4 Proof by Construction

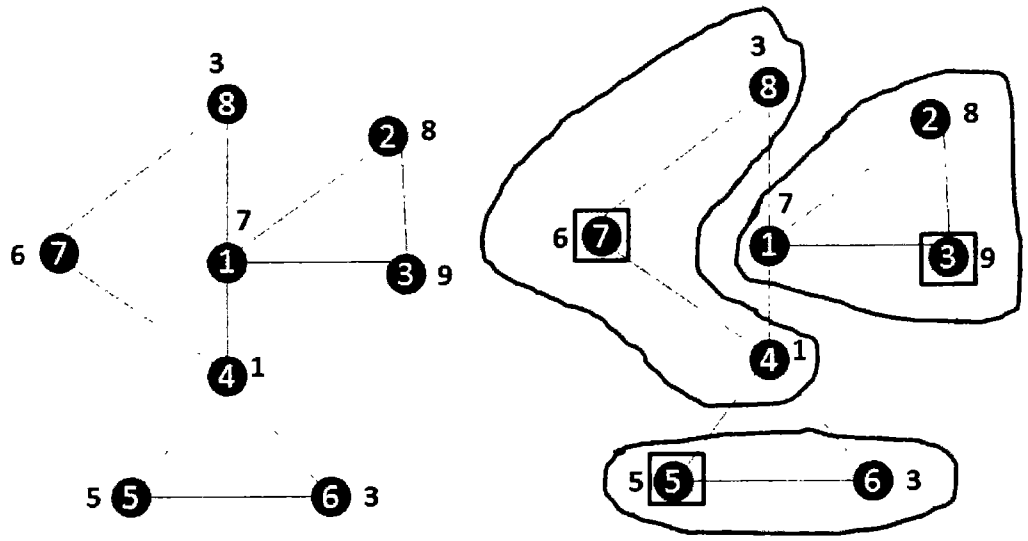


Figure 5: A graph G and clustering of G, squared nodes belongs to the maximal weighted independent set.

6.4.1 Proof Description:

Initially each node broadcasts its transmitting power to its neighbors. In step one node 3, 5 and 7 broadcasts clusterhead message to their neighbors and quit the execution. By the end of the same step node 2 and 1 receive Ch(3) message, node 6 receive Ch(5) message, node 8 receive Ch(7) message and node 4 also receive Ch(7) message.

In step 2 node 1 and 2 send Cnct(1,3) and Cnct(2,3) message to its neighbors similarly node 6 send Cnct(6,5) message and quit the execution. Node 4 and node 8 cannot yet join any cluster because for node 8 their is a node in its neighbor whose transmitting power is bigger then node 7 so node 8 keep waiting for the message from node 1. Same is the case with node 4, their is a node in its neighborhood whose power is bigger then node 7.

By the end of second step node 8 and 4 receive Cnct(1,3) message which means node 1 already connected to cluster whose clusterhead is 3. On that note, node 8 looks for other options, it will send Cnct(8,7) message because $p(7) > p(8)$ and quits the execution. Finally we have node 4, by the end of step two node 4 also receive Cnct(1,3) message and looks for other option. Node 4 will send Cnct(4,7) message and quits the execution of algorithm, eventually we have disjoint clusters.

There are situations in which a node left alone i.e. all of its neighbors with bigger transmitting

power (then its own) connect with other clusters in that case that particular node will make its own cluster by declaring itself clusterhead, but if even any single node exist in its neighborhood whose transmitting power is greater then its own and which still not joined any cluster will be its clusterhead.

6.5 Complexity of Intercluster Edge Selection

Few applications demands a representative edge (among different edges) between clusters. These preferred edges are selected for inter-cluster communication. We can chose these edges arbitrary in a straightforward ways i.e. by broadcast and convergecast. For choosing representative edge between clusters we can apply distributed depth first search from the root/center of cluster each internal node forwards search message. Each node knows its residence i.e. the cluster Id to which it belongs therefore it can identify such edges. Each external nodes uses piplined broadcasts convergecast method to upcast such information to the root then root selects one of these edges and broadcasts it to all. In the synchronous model time required for computing this on a tree T is $Depth(T) + k$ and $O(n)$ message for convergecast. [2]

7 Complexity Analysis And Correctness

The correctness and complexity analysis of Proposed algorithm is discussed in this section, We have also compared our technique with couple of techniques and provided results.

7.1 Correctness of Proposed Clustering Algorithm

Nodes that forms an independent set will send a clusterhead message and vice-versa therefore by the definition of independent set, no two neighbors can be clusterhead.

Theorem 7.1. *Two clusterheads will never be neighbors.*

Proof. For contradiction, let two clusterheads u and v are neighbors and suppose $p(u) > p(v)$ then by definition of maximal independent set v can not be a clusterhead because v has a neighbor u whose transmitting power is greater then v and u has not joined another clusterhead. (u has already sent Ch(u) message). □

By using this algorithm we can generate partition of graph into $k < n$ clusters. By applying the cluster generation procedure to wireless network G, a function is required to assign to each wireless

node v i.e. $\tau : V \leftarrow \{1, ..., 2k\}$ this particularly assigns an integer no. as defined;

$\tau(v) = 2i - 1$ if v is a C_i, s clusterhead

$\tau(v) = 2$ if v is a C_i, s ordinary node.

From the function defined above we can say that for each cluster C_i, $i = 1, ...k$ with its clusterhead v is $\tau(v) = \tau(u) - 1$, where $u \in C_i \setminus \{V\}$. This function indicates the time in no. of steps that v required to decide its role.

Statement 1: All the neighboring nodes u (if exist) of a clusterhead v such that $p(u) > p(v)$ are ordinary nodes such that $\tau(u) < \tau(v)$.

Proof. Suppose C_i, for $i = 1, ..., k$ be the cluster of v. If we consider that among all the nodes that are not yet assigned to a cluster, node v has the biggest transmitting power. It means that all the nodes u in its neighborhood with $p(u) > p(v)$ are; ordinary nodes and they belongs to any other cluster C_j with $j < i$. If $wp(u) > p(v)$ and u is a clusterhead in v, s neighborhood then v could never be a clusterhead and in that case v would be member of u, s cluster. Secondly if any ordinary node u with $p(u) > p(v)$ doesnot belongs to any cluster yet then it can declare itself a clusterhead and v become its member, by the definition of τ it implies that $\tau(u) < \tau(v)$. □

Statement 2: Each node v of the network broadcasts exactly one message within $\tau(v)$ steps. Where $\tau(v)$ is a function assign to each node and $\tau(v) = l < 2k < n$.

Proof. This proof is based on induction on $\tau(v) = l < 2k < n$ where k is number of clusters.

Bases: When $\tau(u) = 1$

It mean that the node is clusterhead in the cluster and will send the message to neighbors nodes. All the nodes u sends either Ch or Cnct message within l steps. For $\tau(v) = l + 1$ step we have two cases;

- v is clusterhead. Then from Proposition 1 we know that all the neighbors u of v with bigger transmitting power then v are ordinary nodes. So by the induction hypothesis in the lth step they sent a $Cnct(u, t)$ message to v. If u is the last node in v,s neighbor with transmitting power bigger then v that send Cnct message then after executing on receiving $Cnct(u, t)$ message node v broadcasts $Ch(v)$ message by $l + 1$ step.
- v is ordinary node. In this case since u is the clusterhead of v and v send a $Cnct(v, u)$ message to u after getting $Ch(u)$ message from u. The u (clusterhead) send the $Ch(u)$ message in l steps and v send $Cnct(v, u)$ message in next step i.e. $l + 1$ which prove the proposition that each node send exactly one message in $\tau(v)$ steps.

□

Sub result 1. Algorithm terminates within $2k$ steps, being k the cardinality of the maximal weighted independent set found.

Proof. Whenever any node sends message of any type, it quits the execution of algorithm. In an example 6.3 algorithm terminates in exactly $3 < 2k = 6$ time steps. The time complexity of this procedure depends upon the number of clusters created by the algorithm.

□

Sub result 2. The algorithm cluster construction takes $O(n)$ messages in the worst case.

Proof. It is clear from statement 2 that n number of messages transmitted. □

Sub result 3. Each node belongs exactly to a cluster.

Proof. There is no overlapping between nodes, i.e. each node joins only one clusterhead. Each node is either a clusterhead or member of a cluster. The set of all clusterheads forms an independent set. In any case each node belongs to a cluster, whether its cluster head or not. If a node u broadcasts a $Cnct(u, t)$ message, t is neighbor of u, then u belongs to a cluster whose head of cluster is t. □

7.2 Comparison of Proposed Technique with Previous Techniques

We have compared our proposed technique with two popular cluster formation techniques and achieved efficient results by simple construction. Figure 6 shows the network with 15 nodes, we show the output of three algorithms in the same topology network. Figure 7 shows minimum Id cluster formation in which node having minimum id is chosen as clusterhead. When we use Minimum Id clustering algorithm, the clusters only have clusterhead and gateway nodes. Many Ch have node degree 1 which is also in the case of Maximum Connectivity clustering algorithm node 2, 5, 7 has degree 1. In our proposed scheme there is only one Ch with node degree 1.

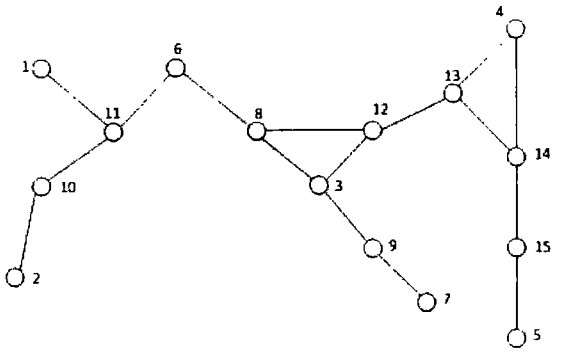


Figure 6: Network Topology

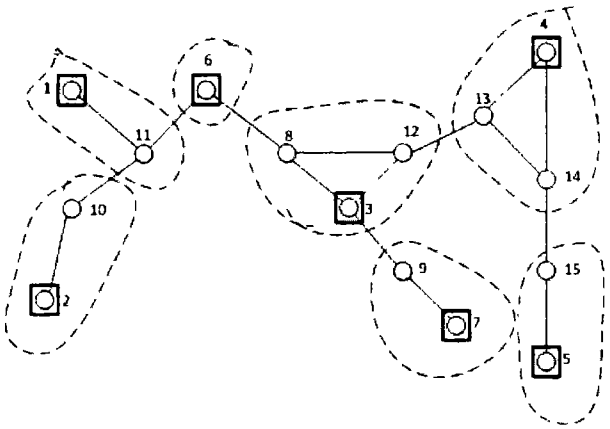


Figure 7: Minimum Id based Cluster Formation

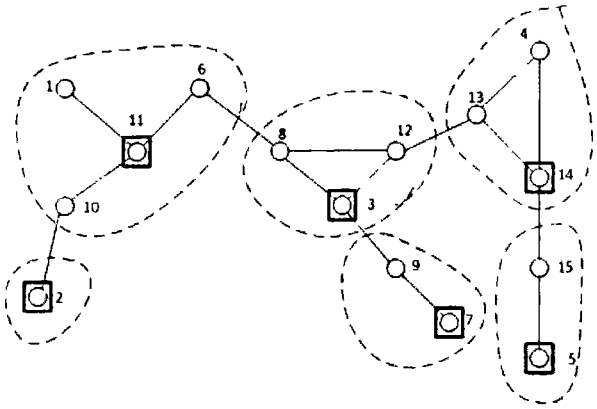


Figure 8: Highest Connectivity based Cluster Formation

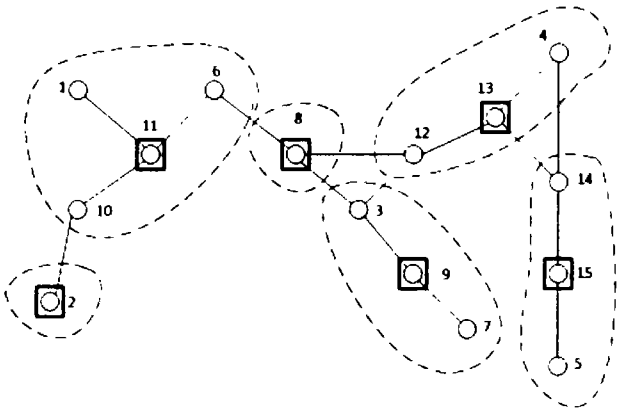


Figure 9: A Proposed Clustering Technique

For inter-cluster communication gateway node play vital role. The smaller the number of gateway nodes, the more possibility of shortest path of transmission. If we want a route between node 2 and node 7 there are 4 gateway nodes in minimum Id clustering and also with maximum connectivity clustering. If using proposed clustering algorithm we need 3 gateway nodes. So we can obtain shortest path by using proposed algorithm.

Although minimum id clustering algorithm is simple and fast but it results in large number of clusters. The maximum degree algorithm reduces the number of clusters but it has a drawback, when a cluster has too many members throughput of each node will be a big drawback.

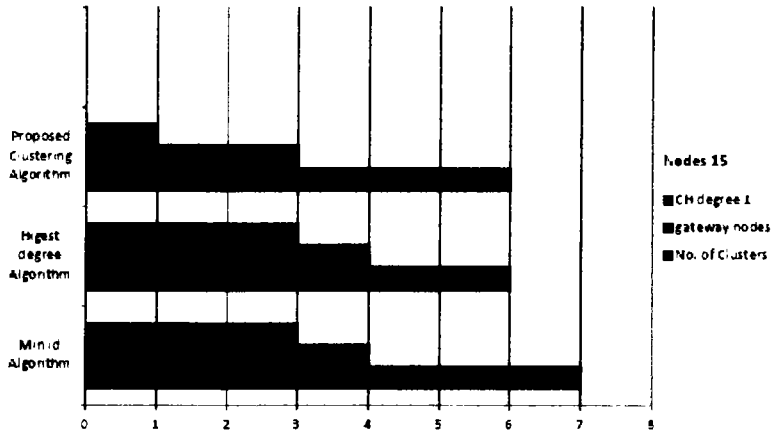


Figure 10: Comparison of Cluster Formation Techniques

This graph shows the comparison result of proposed algorithm with couple of well known clustering techniques.

If we want to compute the transmitting power based clusterhead then we can not use minimum id or maximal degree algorithm in that case proposed technique will be used for cluster formation. Through analysis and comparison of the results of example, we can see that using our proposed algorithm is superior minimum id or highest degree algorithm in many aspects. Even using other attributes of nodes it will solve clustering problem and its correctness is verified through simple example.

8 Conclusion

This thesis presents an efficient distributed solution for partitioning a network into clusters. It addresses the major problem in network and its possible solution. The problem of finding maximal weighted independent set matches the graph partitioning problem so we used one of the technique to solve this problem. The algorithm requires $O(n)$ messages to solve the problem in the worst case whereas the time complexity is proportional to the number of clusters created by the algorithm. Our proposed algorithm balances latency of network and minimizes load by clustering. The performance of this technique can be further investigate using other models like asynchronous, congest.

References

[1] B. Awerbuch and David Peleg., Sparse Partitions,Proc.31st *IEEE* Symp on Found. of Comp. Science, 1990

[2] David Peleg, Distributed Computing: A locality- Sensitive Approach, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000

[3] Nancy A. Lynch, Distributed Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1996

[4] Dr. Stefan Schmid, "Lecture notes: Network Algorithms" Winter term 2013/14

[5] T. Cormen, C. E. Leiserson, R.L. Rivest and C. Stein, "Introduction to Algorithms" MIT Press/McGraw-Hill, Cambridge , MA , 1990.

[6] K.Erciyes, "Distributed Graph Algorithms for Computer Networks", *Springer Publishing Company*, Incorporated 2013

[7] R. G. Gallager, P.A. Humblet and P.M. Spira, "A Distributed algorithm for minimum weight spanning tree", ACM transaction on programming languages and systems(TOPLAS), 5(1):66-77, January 1983

[8] B. Awerbuch, Optimal Distributed Algorithms for Minimum weight spanning tree, Counting, Leader election and related problems, Proceedings of the 9th Annual ACM symposium on Theory of Computation pp.230-240, 1987

[9] B. Awerbuch and R.G. Gallager, A New Distributed Algorithm to find Breadth First Search Trees, IEEE transcation on information theory,33(3) May 1987.

[10] Hand book of graph theory CRC press 2nd edition

[11] R.B.Muhammad, "Lecture notes: distributed algorithms" Creative Common Attribution –Non Commercial- share Alike 3.0 US licensed, 2013

[12] Jefrey Lijffijt, "Realization of unit disk graphs" Seminar graph drawing from Utrecht University department of Informtion and computer science, October 17, 2007

[13] Brent N. Clark and Charles J. Colbourn, David S. Johnson, "Unit Disk Graphs" Elsevier Science Publisher B.V. (North- Holland), Discrete mathematics 86(1990) 165-177

[14] Tapan k Sarkar, Z. ji, K. Kim, A.Medouri and M. Salazar palma. "A survey of various propa-

gation models for mobile communication" IEEE antennas and Propagation magazine. 45(3) 2003

[15] T. S. Rappaport ,"Wireless communications principles and practices" 2nd edition Prentice-Hall, chap 3 pp 70-110, 2002

[16] Er. Neha Sharma, Dr. G C Lal, Study of various indoor propagation models, IJREAS, 1(4), ISSN 2294-3905, 2011.

[17] Stanley L, Aftab Ahmed, Jonathan, M.Graham, Cheryl V. Hinds, L. A. Wahsheh, A. T. Willams and Sundara J D, Empirical channel model for 2.4Ghz IEEE 802.11 WLAN

[18] Peter Steenkiste, Dina Papagiannaki, Wireless Network, Lecture 4: More Physical layer, spring semester 2009.

[19] Dr. Ivica Kostanic, ECE 5221 Personal Communication Systems, Lecture 3: Planning for Coverage in Cellular Systems: Chapter 2.3, spring 2011, Florida Institute of technology.

[20] Hagit Attiya and Jennifer Welch, Distributed Computing: Fundamentals, Simulations and Advanced Topics. John Wiley and Sons, second edition, 2004

[21] James Aspnes, "Notes on Theory of Distributed Systems" CS 465/565: Spring 2014.

[22] N. Ehud Meiri, Reliability of Distributed systems, Lecture 2: Async 2 November 7, 2004

[23] Stefan Schmid. Lecture notes "Distribiuted Computing over communication networks: Vertex Colring" T-labs, 2011

[24] R. G. Gallager. Distributed minimum hop algorithms. Technical Report LIDS-P-1175, Massachusetts Institute of Technology, Cambridge, January 1982.

[25] G.N. Frederickson,"A distributed Shortest path algorithm for a planar network", Information and computation 86, 140-159 1990

[26] S.Moran S. Snir, "Simple and efficient network decomposition and synchronization", Theoretical computer science 243, 271-243 ELSEVIER 2000.

[27] R.B.Muhammad, "Network Algorithms, Applications of Steiner tree and Voronoi diagram" LAMBERT Academic Publishing, 2009

[28] F.Chin and F.H.Ting,"An almost linear time and message Distributed algorithm for minimum weight spanning trees", Proceeding of 1985 FOCS conference, Portland, Oregon, October 1985

[29] E.Gafni, "Improvements in time complexities of two message-optimal algorithm," Proceedings of 1985 PODC conference, Minacki, Ontario, August 1985

[30] B.Awerbuch, B. Berger, L. Cowen and D. Peleg, "Near Linear Cost Sequential and Distributed Constructions of Sparse Neighborhood covers"

[31] B.Awerbuch, A. V. Goldberg, M.Luby, S.A. Plotkin,"Network Decomposition and Locality in distributed computing"Proc. 30th IEEE Symp on Found. of Comp. Science 1989

[32] B.Awerbuch, B. Berger, L. Cowen and D. Peleg,"Fast distributed network decompositions and covers" Journal of Parallel and distributed computing 39,105-114. 1996

[33] B. Awerbuch, "Complexity of network synchronization" J. of the ACM, 32(4):804-823, October 1985.

[34] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. IEEE transactions on communications. 23(3):246-257, 1984

[35] E. Kranakis, H. Sing, and J. Urrutia. Compass routing on geometric networks. In proceedings of 11th Canadian Conference on Computational Geometry (CCCG 1999), pages 52-54, Vancouver Canada, August 1999.

[36] P. Bose, P. Mortin, and I. Stojmenovic. Routing with guaranteed delivery in ad hoc wireless networks. In proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DailM'99), pages 48-55, Seattle, USA 1999.

[37] F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile and ad hoc routing. In proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DailM'02), pages 24-33, Atlanta, USA, September 2002.

[38] X. Y. Li, G. Calinescue, and P. J. Wan, Distributed construction of Planar Spanner and routing for ad hoc wireless networks. In proceedings of 21st Annual Joint Conference of the IEEE Computer and Communications societies (INFOCOM 2002), volume 3, pages 1268-1277, New York, USA, 2002.

[39] S. Basagni, "Finding a Maximal weighted independent set in wireless networks" Telecommunication Systems 18:1-3, 155-168, 2001

[40] S. Basagni, I. Chlamtac and A. Farago, "A generalized Clustering algorithm for peer to peer networks" in: Proceedings of Workshop on Algorithmic Aspects of Communication (satellite workshop of ICALP), July 1997

[41] M. Chatterjee, S. K. Das and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks", IEEE Journal of Clustring Computing Vol.5(2), 193-204, 2002

[42] F.G.Nocetti, J.S. Gonzalez and I.Stojmenovic, Connectivity based k-hop clustering in Wireless Networks." Telecommunication Systems 22:1-4, 205 -220, 2003

[43] Yu-Xuan Wang, F. S. Bao, " An Entropy based Weighted Clustering Algorithm and its Optimization for Ad hoc Networks" 3rd IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Communication, 2007.

[44] G. Zhu, Xi Jiang, Chun Wu and Zhiqiang He, " A Cluster Head Selection Algorithms in Wireless Network based on Maximal Weighted Independent Set" in Ubiquitous Information Technologies and Applications (CUTE), 2010 Proceedings of the 5th International Conference, IEEE, 1-6, 16-18 Dec. 2010

[45] V. Kumar and R. K. Yadav, " Modification in Weighted Clustering Algorithm for Faster Clustering Formation by Considering Absolute Attributed of Mobile Nodes and Greedy Method for Role Selection of Mobile Nodes in MANET" International Journal of Science Engineering and Technology IJSET, Vol.4(12), 572-576, Dec 2015