CONVERGECASTING IN WIRELESS SENSOR NETWORKS

by

Valliappan Annamalai

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

December 2002

CONVERGECASTING IN WIRELESS SENSOR NETWORKS

by

Valliappan Annamalai

has been approved

December 2002

APPROVED:

_____, Chair

_____

_____

Supervisory Committee

ACCEPTED:

_____

Department Chair

_____

Dean, Graduate College

ABSTRACT

A Wireless Sensor Network (WSN) consists of sensors implanted in an environment for collecting and transmitting data regarding changes in the environment based on the requests from a controlling device (called a base station) using wireless communication. WSNs are being used in medical, military, and environment monitoring applications. Broadcast (dissemination of information from a central node) and Convergecast (gathering of information towards a central node) are important communication paradigms across all application domains. Most sensor applications involve both convergecasting and broadcasting. The time taken to complete either of them has to be kept to a minimum. This can be accomplished by constructing an efficient tree for both broadcasting as well as convergecasting and allocating wireless communication channels to ensure collision-free communication. There are several works on broadcasting in multi-hop radio networks (a.k.a. ad hoc networks) which can also be used for broadcasting in WSNs. These algorithms construct a broadcast tree and compute a schedule for transmitting and receiving for each node to achieve collision-free broadcasting.

There has not been much work on convergecasting in multihop networks or in wireless sensor networks. One way to perform convergecasting is to use a tree constructed by a broadcast approach and then assign a schedule over this tree. This schedule might increase the time taken for convergecasting. This thesis focuses on developing a tree construction and schedule assignment algorithm for convergecasting in wireless sensor networks. Two algorithms (pipelined CTCCAA and non-pipelined CTCCAA) are proposed in this thesis. These algorithms minimize the time taken for convergecasting compared to some of other tree construction algorithms. Simulation also show that these trees also minimize time taken for broadcasting.

To my parents, Visalakshi, Lakshmi Narayana, Bharath, Dinesh, Rajesh, Ramanan and Vijay

ACKNOWLEDGMENTS

Dr. Gupta, Dr. Sen and Dr. Martin were a constant motivating and guiding source. The research was supported in intellectual and motivating ways by all of them. They were also good critics and made me learn from my mistakes. I also thank the NSF grants ANI-00196156, ANI-0086020, ANI-0123980, and DGE-9870720 for supporting me during the course of my research.

My peers in MCN lab were very supportive. I thank Aarthi, Sriram, Suresh and Vikram for their time and patience when I really stressed them out.

A special thanks goes to my parents, my sister, Lakshmi Narayanan, Rajesh, Bharath, Dinesh and Ramanan without whom I would not have undertaken this research. Their persuasion and constant motivation helped me to get the results which this thesis puts forth

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER 1

Introduction

The advances in the micro-electronics industry and wireless technology have led to the development of sensors which can be used for accurate monitoring of inaccessible environment. A sensor is a device, that can be controlled and queried by an external device to detect, record, and transmit information regarding a physiological change or the presence of various chemical or biological materials in the environment [2]. A collection of such sensors form a network that can be used for efficient monitoring of health, environment, military etc. [1]. These sensors use the wireless medium to communicate amongst themselves and with a central node capable of controlling this network. We call such networks wireless sensor networks.

The communication in a wireless medium can be classified as infrastructure based communication and ad-hoc communication. In case of infrastructure based communication the nodes in the network make use of a preexisting infrastructure to communicate amongst themselves. Each entity in the infrastructure has a coverage range. This is fixed based on entities transceiver characteristics. These entities have a very small overlapping coverage region and they also keep track of all the nodes that are in the coverage region. When a node in the network wants to send data it sends it to the entity which covers it and the entities in the infrastructure takes care of routing the message to the destination. The en-

tities in the infrastructure are static and the nodes in the network may be mobile or static. To prevent interference in the communication between the nodes and their entity each node is allocated different channels for sending the message. Whenever a node enters an entities coverage region the entity allocates a channel for the new node. In an infrastructure based communication the pattern of communication is clearly defined. This aids in unnecessary wastage of power. In ad-hoc networks each node in the network needs to know its neighbors and the medium is a broadcast medium. Therefore if a node wants to communicate with another node it has to acquire the channel and find out the route from itself to the destination and then sends the message. The problem with this communication is each node in the path has to acquire the channel before transmitting it to the next node in the path and the possibility of packet corruption is high. If an IEEE 802.11 network is used RTS/CTS mechanism is used for acquiring the channel. This mechanism also increases reliability in the network by also providing a retransmission mechanism. This communication pattern is more useful in networks where the communication is not deterministic. But the higher amount of power is used up for communication in and ad-hoc network.

A wireless sensor consists of a small processor, memory, power, sensing and transceiver units. Additionally a sensor can have location finding system, mobilizer and a power generator which are application dependent sub-units [1]. Major part of power consumption by a sensor is to run the transceiver circuitry. As transmission range of a sensor increases the power consumed by the transceiver also increases. So it is better to keep the transmission range as small as possible. This implies if two nodes are not in the transmission range of each other they have to make use of intermediate nodes (multihop) to exchange information.

Sensor nodes have processing capabilities which helps them process the data they

have collected instead of sending the raw data they have collected. The following are some of the properties of a sensor network

- The sensor nodes in the sensor networks can be several orders of magnitude higher than the nodes in an adhoc network.
- Sensors are prone to failure
- Sensor nodes are densely deployed
- Sensor nodes mainly use broadcast paradigm where as most ad hoc networks are based on point to point communication.
- Sensors nodes are limited in power, computation capabilities and memory

Since the sensors are close to each other the amount of power consumed for multihop communication is less compared to single hop communication. Multihop communication also helps in signal propagation effects experienced during long distance communication [1]. There are various kinds of sensors available. These sensors can be used for continuous sensing, event detection, location sensing and local control of actuators.

Sensors can be used in military applications for monitoring friendly forces, Battlefield surveilance, reconnaisance of opposing forces and terrain, targetting, battle damage assessment, nuclear, biological and chemical attack detection. There are environmental applications for forest fire detection, flood detection and precision agriculture. These sensor networks can also be used for environment control in office buildings, interactive museums, managing inventory control and vehicle tracking.

There are lots of factors that influence sensor networks. Some of them are as follows

- Fault tolerance

- Scalability
- Production cost
- Hardware constraint
- Sensor network topology
- Environment
- Transmission media
- Power consumption
- Security

Fault tolerance is nothing but the network should not fail if some of the sensors in the network fail. The sensors which are working properly must reconfigure themselves to form a network. As sensor die they have to be replaced. So new sets of sensors are strewed in the area of interest. The existing network must be capable of detecting that and should be able to include them into the existing network and still be functional. As mentioned earliar sensors are deployed in huge numbers for most of the applications. Therefore the production cost of these sensors must be as low as possible to make such networks feasible. The size of sensors is a big constraint on the hardware that can be included in a sensor. The topology of a sensor network keeps changing (i.e.) as nodes die and new nodes are added the sensor network must be functional. These sensors are usually placed in hostile environment. So these sensors must be capable of working in such extreme conditions. As mentioned earliar the amount of power these sensors have is less therefore the power consumption for data collection and data transmission must be kept low. The data collected by such sensors is

highly critical so the data must be collected and transmitted in a secure manner. Since these sensor are easily accessible they can be compromised easily.

A set of such sensors are strewed in a area that needs to be monitored. A base station is used for controlling and collecting information from these sensors. Since these sensors are strewed each sensor does not know who its neighbors are. These sensors periodically sense data and there is no mobility in the network. As mentioned earlier it is better to construct a network for the sensor nodes to send information to the base station. If each of these sensors collect and transmit data individually to the base station they would be wasting a lot of power. To prevent this high loss of power a tree is constructed and data from the leaf nodes is sent towards the base station through the intermediate nodes. The tree structure aids in avoiding wastage of energy at nodes that are farther away from the base station. The intermediate nodes collect data from the nodes that are closer to them but farther away from the base station than itself. Then they transmit this data to the next higher level of nodes. This pattern of communication is called convergecasting. In convergecasting there will not be any compression of data because data collected at each node is important.

Many sensor applications require broadcasting and/or convergecasting. Broadcasting is the process of information dissemination from a node in the network to all other nodes in the network. Retinal prosthesis of [19] is one such application. Convergecast is the collection of data sensed at each node in the network towards a central node in the network. But convergecast is usually preceeded by broadcast, or they are interleaved between each other. An example of this communication pattern is environment monitoring in which sensors embedded near the area of interest collect and transmit data to external monitoring device based on the query broadcast by the monitoring device.

Collisions occur in a wireless network, when multiple nodes simultaneously transmit

to the same node over the same channel or a receiver is in the transmission range of another communication taking place over the same channel. Such collisions waste resources (e.g. bandwidth and energy) as well as increase data latency and hence they are undesirable. For broadcast and convergecast to work in a collision-free manner we have to construct a tree and allocate a schedule that specifies for each node in the network the time-slots in which it will receive data from other nodes and the time slot(s) in which it'll send data to other node(s). This schedule is assigned for a particular duration of time and it gets repeated for each such time durations. This form of allocating time-slots and avoiding collision is called TDMA channel allocation. Instead of all nodes contending for the channel they use the channel in a synchronized manner thereby increasing reliability and this is desirable for real-time applications. Each application will have an upper limit on the time within which it has to collect and/or disseminate data. The number of time-slots we can allocate and the duration of time-slots is constrained by this time limit which inturn depends on the tree constructed.

1. System Model

The system consists of n sensors placed randomly in the environment that is being monitored. These sensors are controlled by the base station (with node number 0) placed away from the environment of interest. Each sensor has an omni-directional antenna. The communication medium is homogeneous in nature. The sensor nodes, including the base station in the network have equal transmission and reception range (say d). Each node in the network has atleast one neighbor in the network. All sensors have finite power and they have finite amount of memory. The sensors switch on their circuitry when they transmit or receive information. They also continuously sense data and the amount of data sensed by

each node is also constant.

2. Assumptions

We also assume that the base station has complete information about the location of the nodes and all the other nodes in the network do not have knowledge regarding its neighbors. The data collection starts at the leaf nodes and propagates towards the base station. There is no mobility in the network. All nodes sense equal amount of data. Most sensor applications make use of the ISM band which compels them to use orthogonal codes or frequency hopping sequence for communication. Therefore we also assume that the sensors are capable of using orthogonal codes if they are available.

3. Thesis Overview

Here I've proposed two algorithms for tree construction in wireless sensor networks that work well for application that use convergecasting pattern of communication. In chapter 2 we explain in detail about convergecast communication pattern. Chapter 3 talks about some of the work done in sensor network data aggregation. Chapter 4 talks about the algorithms(CTCCAA) for tree construction for convergecasting communication pattern. Chapter 5 compares the tree constructed by the CTCCAA with the tree constructed by the centralized algorithm of [3].

CHAPTER 2

Convergecasting

Convergecasting is the process of collecting data from a set of nodes by a central node (also known as root or base station) in the network. As mentioned in [1] multihop communication is better than singlehop communication so data collection is usually done by constructing a multihop network. The network is in the form of a tree and is centered at the root node. Once the network is set up each node sends the data it has collected to its neighbor that is closer to the root node. When data is sent in such a manner each and every intermediate node adds its own data and forwards it to its neighbor closest to the root node. The node that transmits is called the child node and the node that receives the data is called the parent node. Usually the root node request for some information from all the nodes in the network by transmitting a broadcast packet. The data collection starts at the nodes that are furthest from the root node. The child node of each node in the network transmit the data they have collected to their parent which inturn transmit the data that they received from their children along with the data they collected to their parent. This goes on till it reaches the root node. In case of a wired network two child nodes can share a bus that is connected to their parent. When these children try to transmit to their parent collision can occur. If collision occurs in a wired network the transmitter detects it, does a random backoff and then it retransmits and this increases the reliability in the network. But in case

of a wireless network the medium is a broadcast medium because of this the transmitter will not know if a collision occurs after transmission. If the receiver is waiting for a packet and does not receive it, then it send a request for retransmission to the transmitter. This is needed to make the network reliable. Energy is consumed during every retransmission. In a sensor network the sensors are energy constrained therefore such retransmission are undesirable. These sensors collect critical data because of this the reliability offered by such network has to be high.

Collision-free communication can be achieved by creating a schedule for the entire network. This schedule is assigned for a particular duration of time and it gets repeated for each such time durations. The schedule specifies time slots (time quantum) in which each node will send or receive data without causing any collisions. This schedule is valid till there is no change in the network. In case a node moves out or it dies(looses its power) then the schedule is reallocated. This form of allocating time slots and avoiding collision is called TDMA channel allocation. Instead of all nodes contending for the channel they use the channel in a synchronized manner. Reliability in such a synchronous network is high. Such high reliability is needed for real-time applications. Each application will have an upper limit on the time within which it has to collect or disseminate data. The number of time-slots we allocate is constrained by this time limit. An example of convergecast tree and schedule in a WSN is given in Figure 1 with the schedule assigned for each node. Node 0, the basestation, is the root node, nodes 9,10,11,12,13,14 and 15 are the leaf nodes, and all other nodes in the network are the intermediate nodes of the convergecast tree. The collection of data starts at the leaf nodes and propagates up through the intermediate nodes till it reaches the root node. The slot size is chosen based on the node that'll transmit maximum data in the network. In the sample network the node 6 transmits the maximum

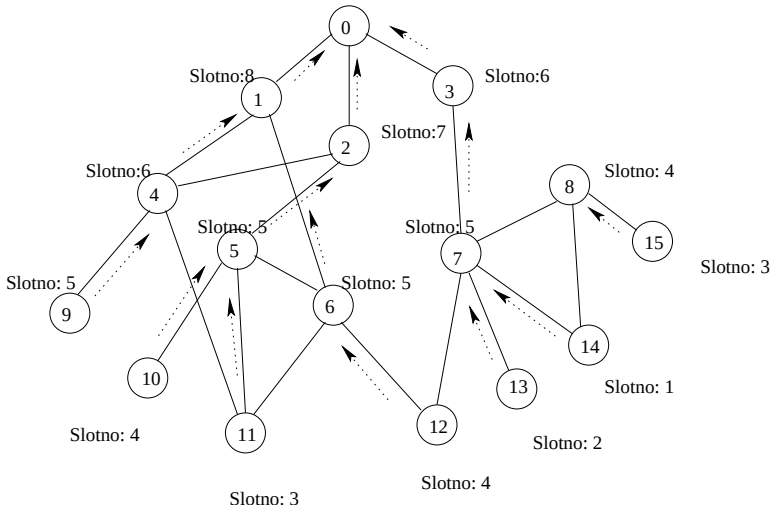

Figure 1. **Sample network from [4] with schedule assigned for convergecasting.**

amount of data (6 units of data). This is fixed as the slot size for the network. The total time taken for convergecasting one set of data collected in the network is the product of total number of slots allocated for nodes in the network with the slot size.

The data collected can be classified into time critical data and those that are not time critical. In case of time critical data the data that is collected cannot be buffered and transmitted. For such applications the data collected at each node is transmitted to the root before they can sense and transmit the next set of data. At any particular point only one wave of data will be traveling in the system (i.e.) each parent waits till it receives the data from its children and upon receipt of data from its children it adds its own data and send it to its parent. Whereas in case of non-time critical data the sensed data can be transmitted to intermediate nodes where it'll be buffered and retransmitted (i.e.) parent nodes do not wait for data from their child nodes but they send what ever data they have buffered. In such a network there might be multiple waves traveling at the same time in the network.

Some applications have an upper limit on the time to collect the data sensed at each

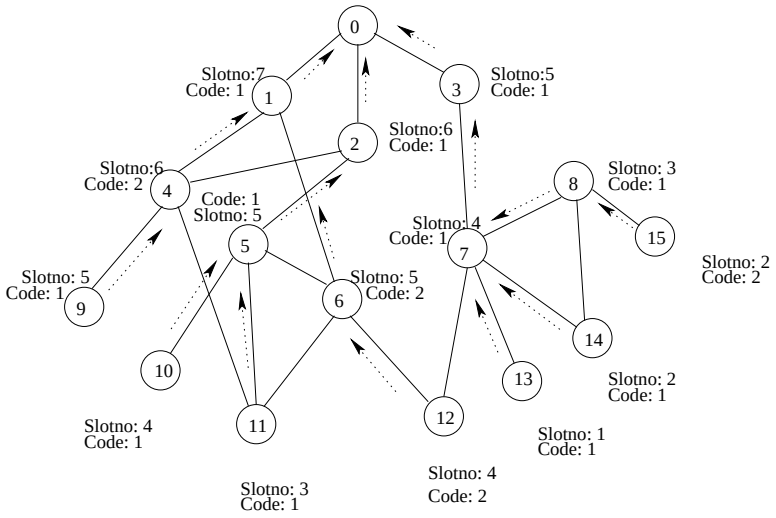

Figure 2. **Sample network from [4] with schedule assigned for convergecasting using multiple codes.**

node in the network. But a convergecast schedule might not adher to this constraint. This can be overcome by using spread spectrum technique (i.e.) orthogonal codes or frequency hopping sequence. This will help in reuse of the same slot by two nodes that are in the transmission range of each of their parents. This helps us in bringing down the total number of time-slots required for communication by dividing independent and non-interfering communications using both time-slots and a spread spectrum technique. Figure 2 illustrates a sample network with schedule assigned for convergecasting using orthogonal codes. In this case the slot size is the same as the previous one but the total number of slots allocated is less than the previous one. This can be further reduced by using more orthogonal codes.

This schedule allocation can be either done centrally or can be allocated in a distributed manner (i.e) each node choosing its parent and allocating itself a slot that does not interfere with any of its neighbors communication with its parent. But as mentioned earlier in the system model nodes in the network do not know about their neighbors and they cannot coordinate together while choosing their slots. Therefore the slots allocation

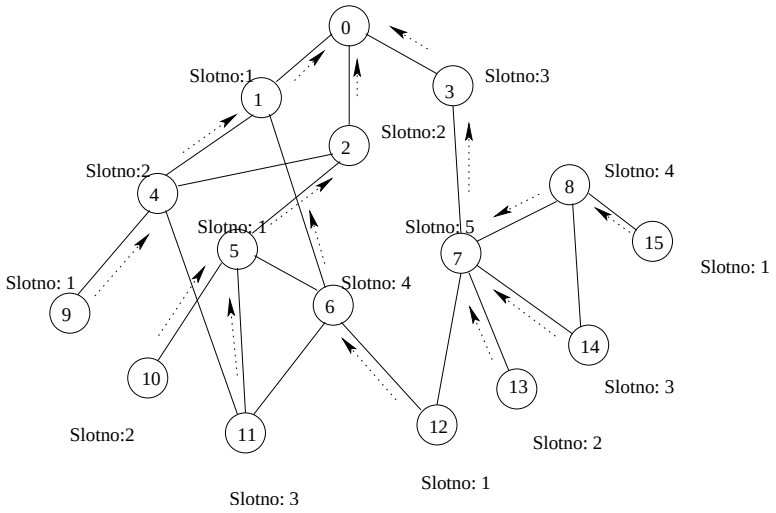

Figure 3. **Sample network from [4] with schedule assigned for non-pipelined convergecasting**

for such a network is done in a centralized manner and then the schedule is broadcast to the nodes in the network.

We can classify convergecasting based on the way data is collected from all the nodes in the network as Pipelined convergecasting (PC) and non-pipelined convergecasting (NPC). In PC the data collection is done one level at a time, starting from the leaf nodes, till it reaches the root node. Figure 1 is a sample network with schedule assigned for convergecasting. Note that in a PC schedule the time-slot say t_1 assigned to a child node should be less than the time-slot say t_2 assigned to its parent (i.e.) the child node sends data to its parent in t_1-th slot of say i-th TDMA frame and the parent node collects the data from all its children and sends it in t_2-th slot of $i+1$-th frame. Figure 3 shows a network with schedule assignment for NPC. In a network with a NPC schedule allocation has nodes that have larger buffering capability and this network cannot be used for collecting real time data.

Formally, the network is modeled as a graph $G = \{V, E\}$. V is the set of nodes

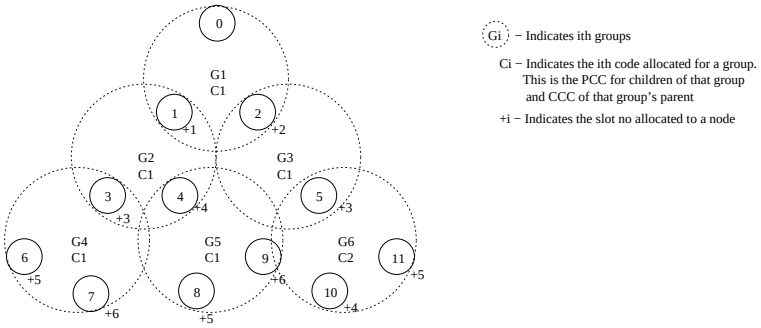

Figure 4. **Example illustrating the constraints.**

$\{0, \ldots, n\}$ in the network and E is set of edges between nodes. We define the edges as follows $\forall x, y \in V$, $\exists (x, y) \in E \iff dist(x, y) \leq d$, where d is the transmission range of sensor nodes in the network and $dist(x, y)$ is the Euclidean distance between nodes x and y. x and y are neighbors of each other. The set of neighbors of a node x is denoted as $neighbors(x)$. For a tree T constructed for convergecasting from a graph G, we call a set $X \subseteq V$, a **group** iff $X = \{y\} \cup Z$ where y is the parent of nodes of Z in the tree T. Proximity of one group's parent p to children of another group leads to interference at p.

Our primary aim is to construct a tree for a given graph G, which will reduce the total time required for convergecasting, for a given set of orthogonal spread spectrum codes (C). We call this problem as *Convergecasting Tree Construction and Channel Allocation Problem* (CTCCAP). A solution to CTCCAP is to construct a tree and to assign to each node i a code called $PCC(i)$ and a slot called $PCS(i)$ to communicate with its parent node. The following are the constraints, that an algorithm should consider in order to compute a valid solution for this problem:

- If the children of one group are in the transmission range of another group's parent then the two groups' children must have different $PCC(i)$ for communicating with their parent if available or they should use different set of $\{PCS(i)\}$ for communicating

with their parents. In Figure 4 node 9 is in the transmission range of group 6. Since there are two codes available group 6 makes use of the code c2 and this allows it to reuse slot 5 used by node 9. If only one code were available then group 6 cannot use slot 5. This is called the co-channel criterion.

- If $dist(nodes1, B) < dist(node2, B)$ then $PCS(node1) > PCS(node2)$ else $PCS(node1) < PCS(node2)$. In each of the groups in Figure 4 the node that is closest to node 0 is the parent. The child nodes of each group are allocated a slot that is greater than the slot used by their parent. This condition is kept in mind while assigning a schedule for PC.

- No two children of a group share the same slot for communication with their parent. But all children of a group share the same $PCC(i)$

- Children for each node are selected based on **proximity criterion** i.e. a node is assigned as a child to the closest possible parent node. This proximity criterion aids in distribution of load experienced during data collection and avoids depletion of energy at a select few nodes in the network.

1. CTCCAP is NP-Complete

To show that CTCCAP is NP-Complete we have to first show that it is in NP and then show that CTCCAP is NP-hard.

1.1. CTCCAP $\in$ NP. In our problem we impose a bound on T then we can check if a given solution satisfies the constraints and has the maximum time-slot as T say k. Given an instance of the problem we can use a certificate which is a graph that contains $|V|$ nodes

and $|C|$ codes and $|T|$ time-slots assigned for communication. The verification algorithm checks if the code assigned and slots assigned adhere to the constraints and also check if the total number of time-slots allotted is less than k (maximum value slots can take). If the assignment satisfies all the conditions then the verification algorithm returns a 1 else it returns a 0 to indicate an invalid assignment. The verification algorithm will certainly take only a polynomial time. So we can conclude that CTCCAP $\in$ NP.

1.2. CTCCAP is NP-hard. We use an Independent set problem (ISP) [5]to show that CTCCAP is NP-hard. We reduce the ISP into an instance of CTCCAP where ISP $\in$ NPC. We have to show that ISP $<=_p$ CTCCAP. For any given instance of CTCCAP we can come up with a grouping of nodes as mentioned earlier in section 1. Let $G = (V, E)$ be an instance of ISP. We reduce ISP into an instance of CTCCAP by representing each node in ISP as a group. We form the complete graph $G^{'} = (V, E^{'})$ where $E^{'}$ is the complement of E. This reduction can be easily accomplished in polynomial time.We have to show that if CTCCAP is satisfiable using k channels then we can find an independent set of size k for the given instance of ISP. Now we do channel allocation for $G^{'}$. Say if we use k channels for $G^{'}$ and then if we take one node for each channel we have assigned, based on the connectivity of nodes, then we can come up with a set with k nodes. These k nodes will form an independent set. Consider the example shown in figure 6. We derive an instance of CTCCAP and do allocation for it and the resultant graph is shown in figure 5. a1,a2,a3 and a4 are the channel used. For each channel choose the node with highest connectivity. We get the set with nodes 1,2,3 and 5. This can be accomplished in polynomial time. This is the independent set for the ISP in figure 6. Therefore we can claim that the problem is NP-hard.

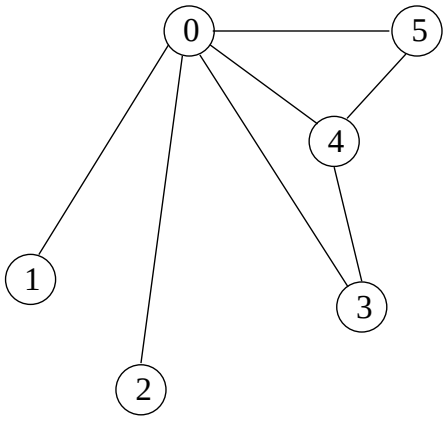

Figure 5. **Instance and solution of ISP**

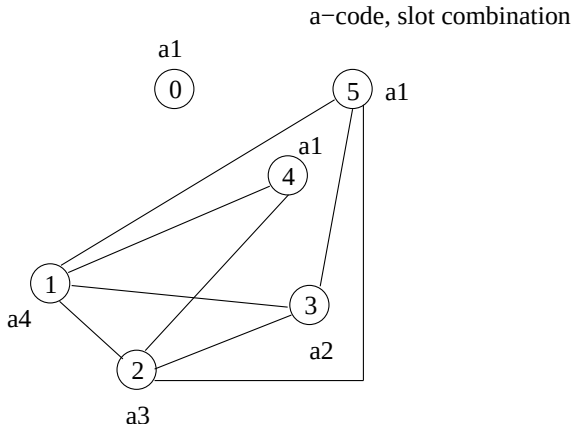

Figure 6. **Instance of CTCCAP**

CHAPTER 3

Related Work

In [3] the authors have proposed TDMA channel allocation schemes for broadcasting in wireless multi-hop radio networks. The algorithm works in a centralized manner. The allocation initially starts at the root node. The root node examines all its neighbors and makes all those nodes which do not have a parent as its children. Then allocates its self a time-slot for broadcasting. Then it chooses children for each of its children and does allocation for each of its children. This is done till all the nodes in the network are covered. It uses the following constraints while choosing a lowest possible time-slot i for node v

- i is not the slot assigned to v's father
- i is not assigned to another neighbor of v's father
- i is not assigned to a father of v neighbor

One simple way to achieve collision-free convergecasting is to compute a new schedule for convergecasting over a broadcast tree constructed by the broadcast approach of [3]. The tree of this approach is not a balanced tree because a node chooses all its neighbors that do not have a slot assigned for broadcasting as its children. But this tree covers all the nodes in the network. If such an unbalanced tree is used then the slots allocated to the

nodes will be of larger size because the slots are assigned based on the node that'll sends large amounts of data. Therefore the time taken of convergecasting will also go up and the nodes will be wasting lots of power in the idle state during their transmission slot after they transmit the data they collected.

In [4] the authors have proposed a TDMA channel allocation scheme for broadcasting in wireless multi-hop radio networks. The root node sends the initial message to all its children. Now these neighbors are the transmitters and their neighbors who havn't yet received this message are the potential receivers. Now the algorithm proposed in this paper (SEA) tries to choose a subset of transmitter that will maximize the number of receivers covered in the next level. This is carried out for each level in the network. The problem with this approach is that all nodes in the network will not be covered by the tree constructed. If this tree is used then data from all the nodes would not be available to the root node.

In [15] the authors talk data aggregation in wireless sensor networks using routing. They propose three schemes that can be used for data aggregation. They are the Center at Nearest Source(CNS), Shortest Path Tree (SPT) and Greedy Incremental Tree (GIT).

In CIT all sources send their data directly to the source which is nearest the sink which sends the aggregated information on to the sink. In SIT the information to the sink is sent to the source along the shortest path and when two paths overlap, the data sent by each source is combined to form the aggregation tree. In case of GIT the source closest to the sink node is connected to the sink and from then on at each step the next source closest to the current tree is connected to the tree. The problem with CIT, SIT and GIT is that each and every node does not have information and they also need a reliable mac layer to function properly. As mentioned earliar if we make use of a reliable mac layer then the amount of energy consumed will be high because of control packets and random backoffs

when there is collision in the network. Therefore none of these approaches can be used for convergecasting.

In [16] the authors propose a chain based binary scheme for data aggregation called PEGASIS. This algorithm starts by constructing a chain of all nodes in the network. The chain creation is initiaded at one of the nodes that is farthest from the sink node. The closest node to this node will be the next node in the chain. Successive neighbors are selected in this manner among the unvisited nodes to form the greedy chain. When the nodes die the chain is reconstructed. All the nodes in the chain have information regarding its predecessor and successor. The data collection is done in cycles. During each cycle one node in the network is choosen as the aggregation point. Say if there are n nodes in the network there'll be n cycles and node i will be the gathering point in cycle i. After n cycles the cycles start repeating again. All the other nodes send the data they have collected to the node that is closest to the sink. They assume that each node compresses the data to such and extent that the data transmitted by each node in a cycle is equal. This yields a good schedule for data aggregation. But it will not be good for convergecasting because the data cannot be compressed and so the slot slot size will be equal to the total number of nodes in the network. This will lead to lot of time and power wastage. Therefore this protocol cannot be used for convergecasting.

CHAPTER 4

Convergecasting Tree Construction and Channel Allocation Algorithm (CTCCAA)

The CTCCAA algorithm presented in this section utilizes a greedy approach for tree construction and channel allocation. The formation of the tree takes place one level at a time and simultaneously allocates channel for communication. The algorithm starts allocation from the root node and proceeds in a breadth first manner. The root node runs the algorithm after termination the root node transmits a broadcast packet with the schedules allocated for each node in the network. The algorithm takes the neighbor list of each node at the current level and starts assigning a channel for each node in the list such that it confers to the constraints and these nodes become part of the next layer of the tree. The algorithm is provided with the list of valid codes that can be assigned, a set of nodes and the position of the nodes. This algorithm can be used for both Pipelined convergecasting and Non-pipelined convergecasting.

1. Pipelined CTCCAA

The algorithm maintains a list called "currentlist" and a list called "nextlist". Initially the currentlist consists of all nodes in level 1 (i.e. the root node) and nextlist will

be empty. For each node (P) in level 1 it chooses the children and allocates a channel for the each of the child nodes to communicate with P. It uses the proximity criterion while choosing children for each node. All these child nodes are inserted into the nextlist. Once allocation for all nodes in the currentlist is done, all nodes in the nextlist are copied onto the currentlist. Now the algorithm chooses each node from the currentlist (i.e. all nodes in current level (level 2)) and chooses children for them and also allocates channel for the child nodes. For convergecasting the slot allocated to a parent node should be greater than the slot allocated to the child node. But in our case after allocation the time slot for a parent node will be lesser compared to the slot allocated to a child node. We reverse the order in which the slots were allocated to the nodes in the network by finding the slot with the highest number and, from that we subtract the slot number allocated to each node to get the actual slot number.

While choosing code and slot for child nodes the algorithm has to keep in mind the co-channel criterion. We can find a set of interfering neighbors (IL_k) for a group which has node k as its parent based on co-channel criterion. IL_k can be defined as

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

Each parent chooses a $PCC(i)$ for its children by choosing the code least used by its interfering neighbors. This is carried out in the choose-code function of the algorithm. Once the parent chooses the $PCC(i)$ for its children it has to choose a slot that can be allocated to its children. This slot has to be greater than the $PSC(i)$ of the parent node. This is carried out with the help of choose-slot function. This way we find the allocation of codes that minimizes the total number of slots required for network. Valid codes and slots that the algorithm can allocate start from 1.

The following are the descriptions of the functions and variables used in algorithm in Figure 7.

Neighborlist(N, d): Returns neighbor list information for each node in the network. Takes information regarding all nodes N in the network and their fixed transmission range d.

Choose-slot(S): Returns least value of slot which will not interfere with the communication of nodes in S.

currentList: Array for storing the nodes in the current level.

InterU(a.nl,b.nl): Does the union of arrays only if the a.nl contains a node which is also part of b.nl.

Choose-code(orthCode, parent): This function aggregates the neighbors of the parent node and the neighbors of all child nodes and chooses a code that is least used by these aggregated list of nodes.

currentlevel: Variable which holds the current level of the tree for which assignment is being done.

orthCode: List of orthogonal codes (given as input).

nextList: Array for storing the nodes in the next level.

Union(a, b): Performs $a = a \cup b$. This is used to aggregate all child nodes to form the parent list for the next level.

Sort(currentlist): The sort function takes care of sorting the currentlist in ascending order of number of children each node has.

d: Transmission range (given as input).

N: Set of nodes and their coordinates (given as input).

Each node maintains the following information:

level: Level number i.e. its position in the tree

pcc: Code used for communication with its parent

psc: Slot used for communication with its parent

ccc: Code used for receiving information form children

csc: Set of slots used to receive information from children

nl: A set that contains information regarding neighbors.

Consider the sample network in figure 8 and let us assume that only one code is available. The following are the steps the algorithm takes while assigning a schedule for the nodes in the network. First node 0, which is the root node, runs the algorithm using the information regarding the nodes in the network. It first chooses time-slot 0 for itself then it chooses all its neighbors as its children because there are no other neighbors that are capable of being the parent of these neighboring nodes and also allocates a time-slot for each of its children. In this case node 1 is allocated a time-slot S1 and node 2 is allocated a time-slot S2. The state of the network after this step is given in figure 9. Then it chooses one of its children, say node 1 and it chooses children for node 1. Node 1 has three neighbors nodes 3,4 and 5. Now let us assume that the distance between node 1 and node 5 is greater than the distance between node 2 and node 5. Therefore node 1 chooses nodes 3 and 4 as its neighbors and allocates them time slots S2 and S3 for them respectively. The state of the network after this step is given in figure 10. After this step node 2 tries to select its children from its neighbor list. The only neighbor of node 2 is node 5 and it has not yet been selected as a child. So node 2 chooses node 5 as its child and allocates a time-slot S4. Figure 11 shows the allocation of slots for each node in the network. Then the reversal of time-slots is done to aid in convergecasting. The final slot allocation for the nodes in the network is shown in figure 12.

```
Algorithm: CTCCAA(N,d,type)
begin
    Neighborlist(N, d)
    currentList = base-station

    while (true)
        if (currentList does not have unvisited nodes)
            currentList = nextList
            currentlevel++
            if (currentList == ∅)
                break
            nextList = ∅
            Sort(currentList)
        end if
            Select the node (nnum) from currentList and mark it visited.

        ccode = Choose-code(orthCode, nnum.nl)
        nnum.ccc = ccode
        Union(nextList, nnum.nl)

        for (each n ∈ nnum.nl) // here we find all interfering nodes
                interlist = InterU(n.nl, nnum.nl) // with nnum's children
        end for

        hslot = Choose-slot(interlist) // Choose the next available slot

        for (each n ∈ nnum.nl)
            if (n.psc != 0 and closeness(n, nnum))
                n.pcc = ccode // Code assignment for child
                n.psc = hslot // Slot assignment for child
                if (type == NPC) hslot = Choose-slot(interlist) num.csc array // nnum adds this to
                                //its receiving slot
                hslot++
            end if
        end for
    end while

    if (type == PC) referenceslot = Highestslot(N) + 1 // Here we do the reversal of slots
        for (each n ∈ nnum.nl) // to find the actual slot
            n.psc = referenceslot - n.psc
        end for
    end if end
```

Figure 7. **Convergecast tree construction and channel allocation algorithm for pipelined and non-pipelined convergecasting.**

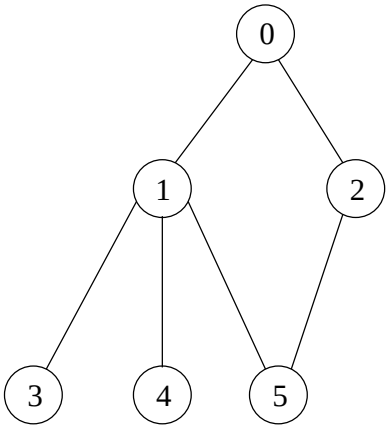

Figure 8. **Sample Network**

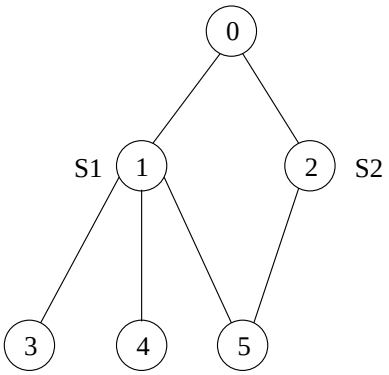

Figure 9. **Network state after slot allocation for nodes 1 and 2**

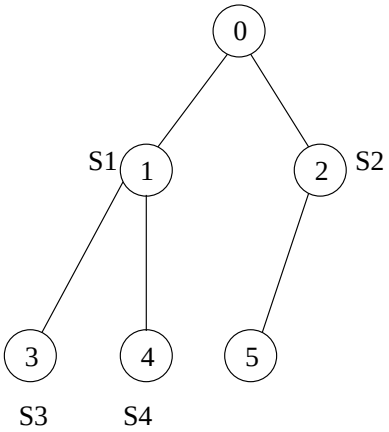

Figure 10. **Network state after slot allocation for nodes 3 and 4**

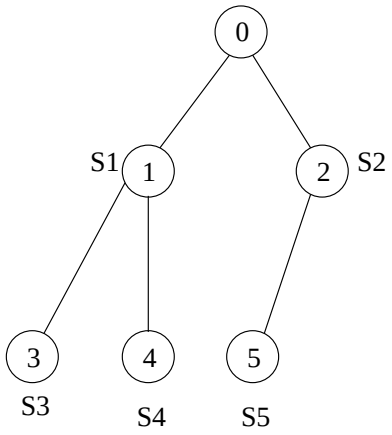

Figure 11. **Network state after slot allocation for all nodes in the network**

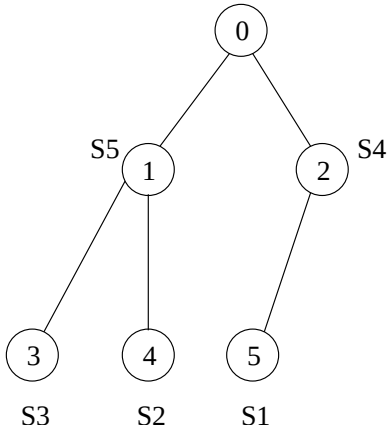

Figure 12. **Network with the final slots allocated to the nodes**

2. Non-pipelined CTCCAA

In non-pipelined algorithm the slot allocation for a child node need not be greater than the slot of its parent. There will be simultaneous communication in independent and non-interfering part of the network. The non-pipelined version of CTCCAA differs from the pipelined when a parent node allocates slot for its children. It first chooses the code that is least used in the non-interfering neighbor set. Then it chooses the least slot, that does not interfere with any of the non-interfering neighbors, for its children for the code which it selected. And the reversal of slots is not done in this case.

Consider the sample network in figure 13 and let us assume that only one code is available. The following are the steps the non-pipelined CTCCAA algorithm takes while assigning a schedule for the nodes in the network. First node 0, which is the root node, runs the algorithm using the information regarding the nodes in the network. It first chooses time-slot 0 for itself then it chooses all its neighbors as its children because there are no other neighbors that are capable of being the parent of these neighboring nodes and also allocates a time-slot for each of its children. In this case node 1 is allocated a time-slot S1 and node 2 is allocated a time-slot S2. Then it chooses one of its children, say node 1 and it chooses children for node 1. Node 1 has three neighbors nodes 3 and 4. Node 1 chooses nodes 3 and 4 as its neighbors and allocates them time slots S2 and S3 for them respectively. After this step node 2 tries to select its children from its neighbor list. The only neighbor of node 2 is node 5 and it has not yet been selected as a child. So node 2 chooses node 5 as its child. It tries to choose the smallest slot that can be assigned to node 5. In this case node 2 allocates S1 to node 5. Figure 14 shows the allocation of slots for each node in the network.

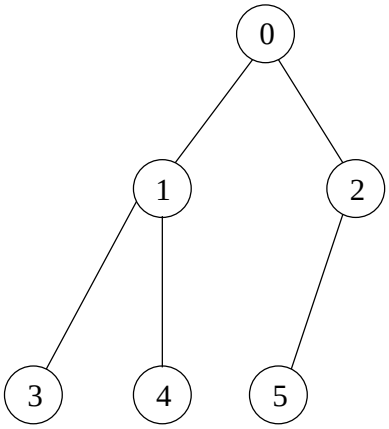

Figure 13. **Sample Network**

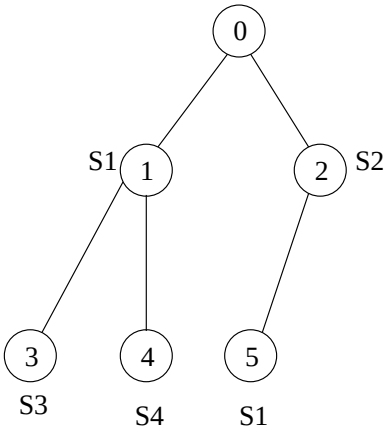

Figure 14. **Network state after slot allocation for all nodes in the network**

3. Theoritical Limits

For a given graph we can define the degree of each node. In our case the degree of a node depends on the transmission distance of each node. Degree of each node is the number of neighbors a node has. From this neighbor information we can find the number of possible parents (Mpi) by choosing all nodes that are closer to the base station than the current node as this node's possible parents. And the maximum number of children (Mci) by finding the number of neighbors that are farther away from the base station than the current node. This can be carried out for all nodes in the graph. We can use this to calculate the theoretical upper limit and lower limit for a given graph.

For any randomly generated network we can calculate the maximum and minimum number of codes required to suffice the graph. The maximum and minimum number of codes required is dependent on the number of possible parents each node in the network has. Once a node becomes a child of a node and is allocated a time slot Ti and code Ci for communication then all the other possible parents have to use a different code to communicate with their children if they use the Ti for communicating with their children. Therefore the maximum number of codes ($Mcci$) required is dependent on the maximum number of possible parents a node can have. Once we know the maximum number of codes necessary with respect to each node in the network we can easily find the maximum and minimum number of codes necessary for the whole network as follows.

$$MCSC = Max(Mcci) \tag{4.1}$$

$$mCSC = Min(Mcci) \tag{4.2}$$

In a similar fashion we can find the maximum and minimum number of slots necessary for a network (MSSC mSSC). Now The maximum number of code slot combination can be derived as follows.

$$MS = MCSC * MSSC \tag{4.3}$$

Similarly minimum number of code slot combination is

$$mS = mCSC * mSSC \tag{4.4}$$

The fartest node in the network determines the maximum number of levels in the network.

$$Maxlevel = farthestnode/txdistance \tag{4.5}$$

Summation of MS for Maxlevels gives maximum number of code slot combinations required summation of mS for Maxlevels gives minimum number of code slot combinations required.

CHAPTER 5

Experiments and Results

The tree creation algorithm proposed here is well suited for networks that involve both broadcast and convergecast pattern of communication. As mentioned in chapter 3 there are some algorithms that are for network construction and schedule assignment but are not efficient interms of latency and energy consumption. To show that the tree construction algorithm proposed here helps in a better schedule assignment, which intern leads to conservation of energy and reduces latency we compare it with the centralized algorithm of [3]. For this purpose we generated scenarios with random placement of nodes for varying node densities. Then we generated the tree for each of the scenarios by CTCCAA and by centralized algorithm of [3] and compared the latency for data collection and dissemination over this trees. For comparison purpose we assume that each node senses 1 *unit* of data. 20 such sets of scenario files for varying nodes densities and for nodes ranging from 20 to 220 was generated. Section 1 explains the way scenario file generation program and sections 2 and 3 we present the performance of pipelined convergecasting and non-pipelined convergecasting. In section 4 we show how tree constructed by our approach is better even for broadcasting.

1. Scenarios

Programs were written to generate random graphs to compare tree constructed by pipelined CTCCAA and non-pipelined CTCCAA with the tree constructed by the broadcast approach of [3]. The generation program generates scenarios based on number of nodes and node density that is given as input to it. It places the nodes in a square grid. For example say if total number of nodes to be placed in a network is n and node density is d, then the total area A needed for placing the nodes in nd. Now from the value of A it sets maximum limit on x axis and y axis as the $sqrt(A)$. Then for generating the random position of x and y value of a node it uses the C programing languages rand function. This function generates a random number between 1 and RAND_MAX (value of RAND_MAX in linux is 2147483647). Now using the x limit and y limit this value is scaled down to x and y position of the nodes as follows

$$x = rand()\%x - limit$$

and

$$y = rand()\%y - limit$$

. The program first places the root node at the point $(0, 0)$. Then for the subsequent nodes the value of x and y are calculated and the node in placed in the network only if the node is the transmission range of one of the nodes in the network. If not the x and y coordinates of the node is again calculated and the above condition is checked. This is carried until the successful placement of the node.

2. Performance of Pipelined CTCCAA

In this section we compare the tree constructed by CTCCAA and centralized approach of [3] for pipelined convergecasting and for broadcasting.

2.1. Implementation. The two versions of CTCCAA algorithm was implemented in C language. The first version takes as input a scenario file and the transmission range of nodes and total number of available codes and constructs a tree and also assigns a pipelined convergecast schedule. The second version of CTCCAA algorithm takes as input a tree and allocates a valid schedule for all the nodes in the tree. The tree for this approach is the broadcast tree generated by the centralized algorithm of [3]. This centralized algorithm was also implemented using C programming language. The program takes as input the scenario file and the transmission range of the nodes in the network and generates an output file that contains the tree constructed by this approach.

Consider the sample network of Figure **??** for which we construct a tree using broadcast approach of [3] and do slot allocation for convergecasting and obtain the network in Figure 15. In the resultant network, data from nodes 3,4 and 5 are sent to node 1 and node 1 adds its own data to the data it received and then sends it to node 0. Since node 2 does not have any children it sends only the data, it had sensed, to node 0. In a TDMA schedule the slots are of equal size. The slot duration must be long enough to accommodate the maximum amount of data collected at any one node in the network. In this case it is node 1 which has to transmit 4 units of data. Therefore we allocate time-slot large enough to hold 4 units of data to each node. So the total time taken for one cycle of convergecasting would be total number of slots used by the network times the slot duration i.e. 20 time units. The power in node 1 drains faster than the other nodes in the network because node

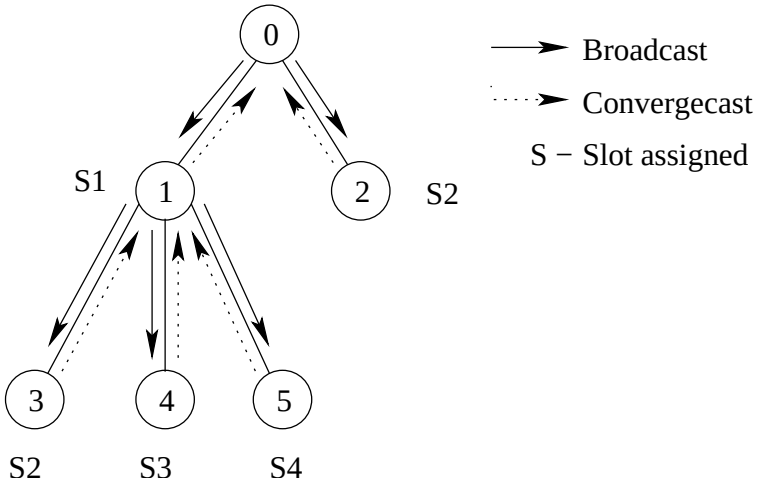

Figure 15. **Slots assigned for convergecasting on a tree generated for broadcasting**

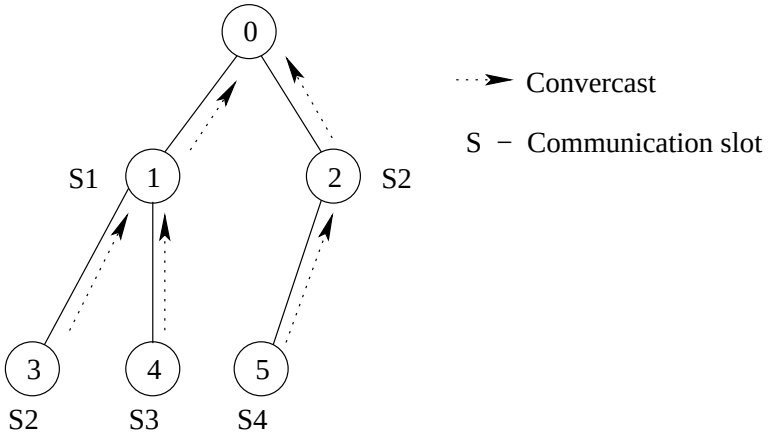

Figure 16. **Slots assigned for convergecasting by CTCCAA**

1 receives more data and has to transmit more data. Other nodes waste a small amount of energy in idle state after they transmit the data that they had sensed for the remainder of the slot duration that was allocated to them.

If the tree is constructed using CTCCAA we get the tree in Figure 16. As stated earlier CTCCAA makes use of proximity criterion which aids in distribution of load experienced during data collection and avoid depletion of power at a select few nodes in the network. Node 1 collects data from its children (i.e. nodes 3 and 4) and node 2 collects

data from its child (i.e. node 5). The total data transmitted by node 1 to node 0 is thrice the amount of data collected at each node and the total data transmitted by node 2 is twice the data collected at each node. Therefore we allocate time-slot long enough to hold 3 units of data to each node. And the total time taken for one cycle of data would be total number of slots used by the network times the slot duration i.e. 15 time units. This is less than the slot duration for a tree constructed by broadcast approach and the load from node 1 is now distributed onto node 2. The total power expended by the system also decreases because the power expended by nodes in idle state is less. From this we can conclude that a new tree construction algorithm is needed for convergecasting.

2.2. Results. Simulations were carried out over randomly generated graphs with node densities varying from 0.1 $node/unit^2$ to 0.8 $nodes/unit^2$. The total duration required for pipelined convergecasting was measured for the trees constructed by both approaches. The total duration required for pipelined convergecasting was measured for the trees constructed by both approaches. From figure 17 to figure 24 shows the result obtained from simulations conducted for pipelined convergecasting. It shows the ratio between total duration for pipelined convergecasting over trees generated by broadcast approach of [3] ($T_{b,c}$) and total duration for pipelined convergecasting using trees generated by CTCCAA ($T_{c,c}$) . From the table 2 we can infer that the tree construction algorithm proposed here works good for node densities above $0.3 nodes/unit^2$.

3. Performance of Non-pipelined CTCCAA

In this section we compare the tree constructed by CTCCAA and centralized approach of [3] for non-pipelined convergecasting.

Table 1. **Comparison of time taken for convergecasting over a tree constructed by CTCCAA with tree constructed by algorithm of [3] on randomly distributed graphs**

Nodes densities $nodes/unit^2$	Performance percentage
0.1	-25 to 50
0.2	-20 to 65
0.3	20 to 80
0.4	5 to 120
0.5	0 to 150
0.6	0 to 150
0.7	0 to 150
0.8	0 to 170

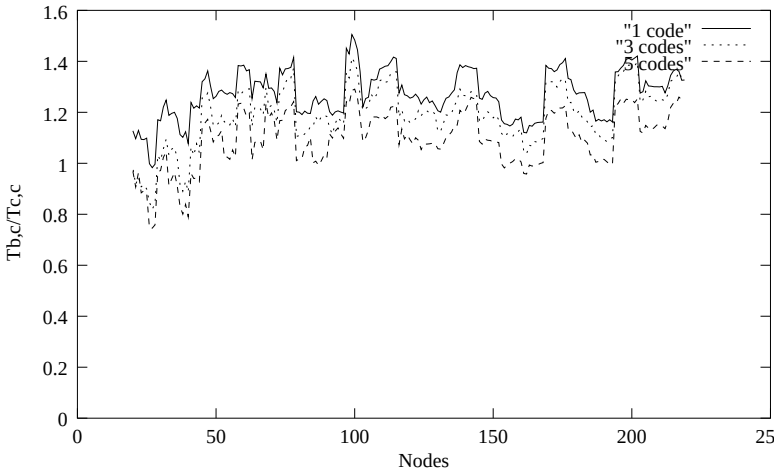

Figure 17. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.1 nodes/unit^2$.

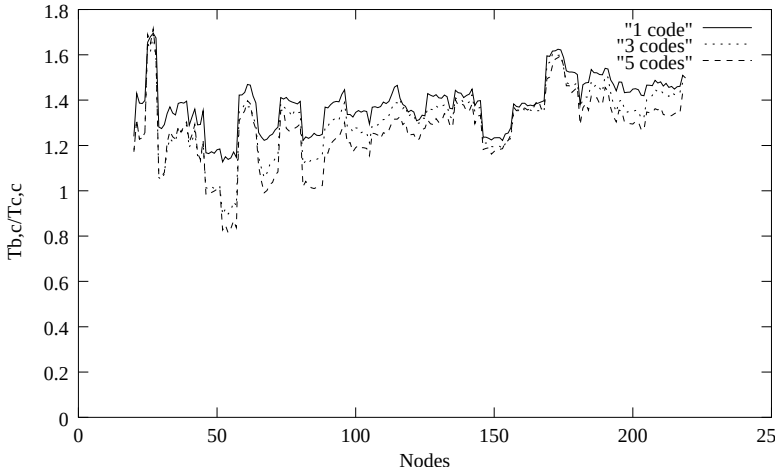

Figure 18. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.2 nodes/unit^2$.

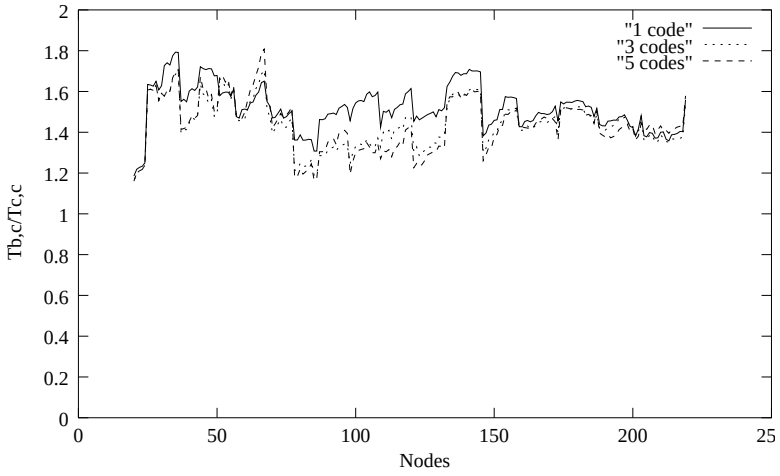

Figure 19. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.3 nodes/unit^2$.

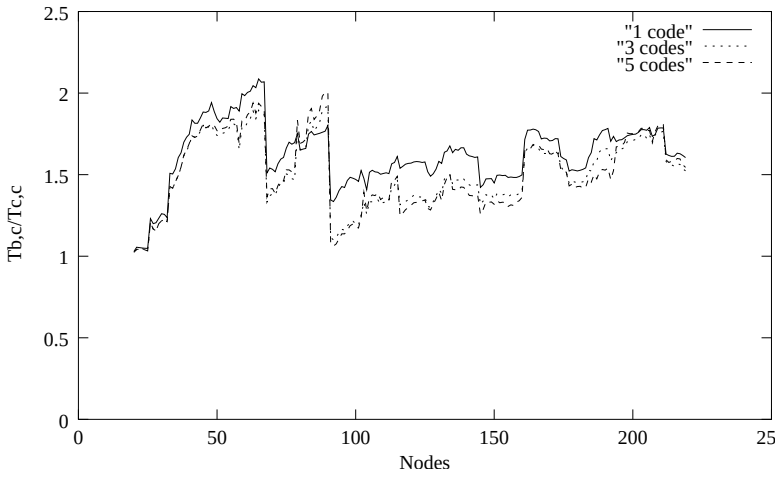

Figure 20. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.4 nodes/unit^2$.

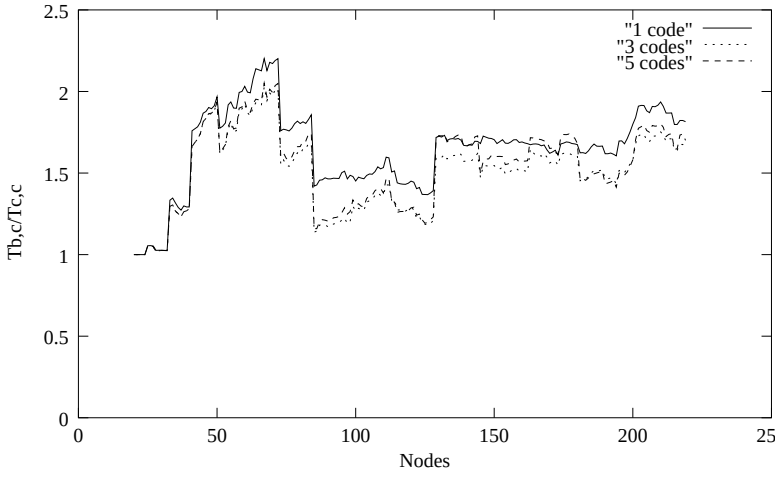

Figure 21. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.5 nodes/unit^2$.

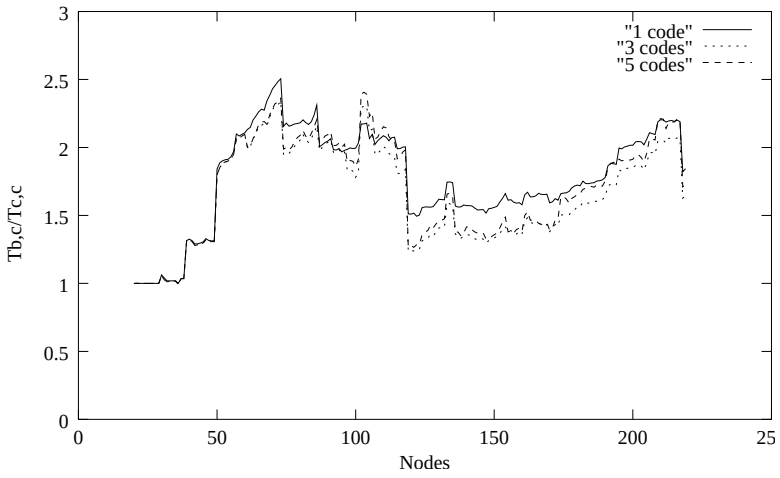

Figure 22. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.6 nodes/unit^2$.

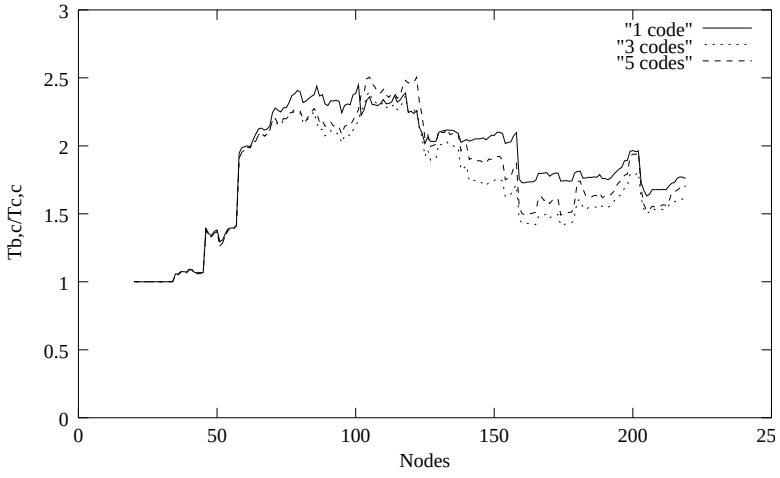

Figure 23. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.7 nodes/unit^2$.

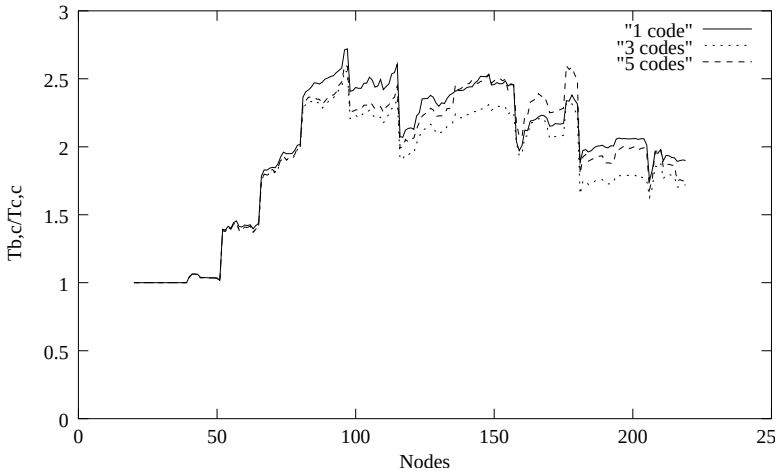

Figure 24. **The ratio of total duration for pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.8 nodes/unit^2$.

3.1. Implementation. The two versions of non-pipelined CTCCAA algorithm were implemented in using C language. The first version takes as input a scenario file and the transmission range of nodes and total number of available codes and constructs a tree and also assigns a non-pipelined convergecast schedule. The second version of CTCCAA algorithm takes as input a tree and allocates a schedule for all the nodes in the tree for non-pipelined convergecasting. The tree for this approach is the broadcast tree generated by the centralized algorithm of [3]. This centralized algorithm was also implemented using C programming language. The program takes as input the scenario file and the transmission range of the nodes in the network and generates an output file that contains the tree constructed by this approach.

3.2. Results. Simulations were carried out over randomly generated graphs with node densities varying from 0.1 $node/unit^2$ to 0.8 $nodes/unit^2$. The total duration required for non-pipelined convergecasting was measured for the trees constructed by both approaches. The total duration required for non-pipelined convergecasting was measured

Table 2. **Comparison of time taken for non-pipelined convergecasting over a tree constructed by CTCCAA with tree constructed by algorithm of [3] on randomly distributed graphs**

Nodes densities $nodes/unit^2$	Performance percentage
0.1	-11 to 90
0.2	-5 to 110
0.3	12 to 142
0.4	0 to 120
0.5	0 to 164
0.6	0 to 190
0.7	0 to 186
0.8	0 to 270

for the trees constructed by both approaches. From figure 25 to figure 32 shows the result obtained from simulations conducted for convergecasting. It shows the ratio between total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] ($T_{b,c}$) and total duration for non-pipelined convergecasting using trees generated by CTCCAA ($T_{c,c}$) . From the table 2 we can infer that the tree construction algorithm proposed here works good for node densities above $0.3 nodes/unit^2$.

4. Broadcasting

To check whether approach the schedule allocation for broadcasting will provide a better solution compared to the centralized algorithm of [3] we construct a tree using CTCCAA algorithm and then assign schedule. Then comparison of the latency for broadcasting is done for both these schedules. The two versions of centralized algorithm of [3] was implemented in C language. The first version takes as input a scenario file and the transmission range of nodes and constructs a tree and assigns a broadcast schedule. The modified version of the CTCCAA algorithm takes as input a network and constructs a tree over these nodes.

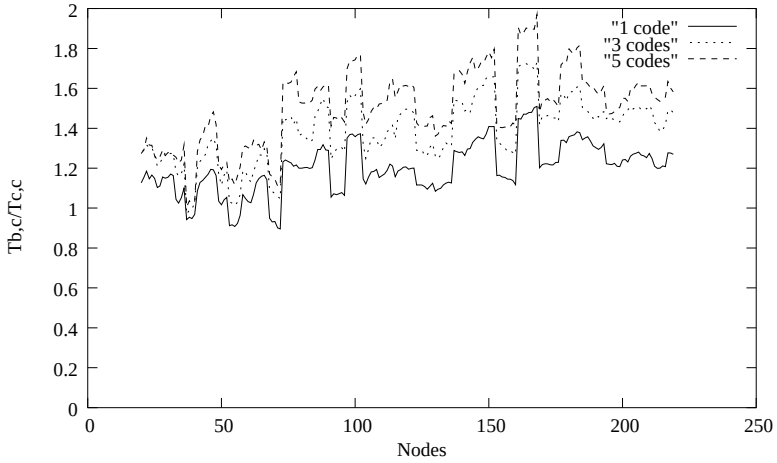

Figure 25. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively on a network with node density** $0.1 nodes/unit^2$**.**

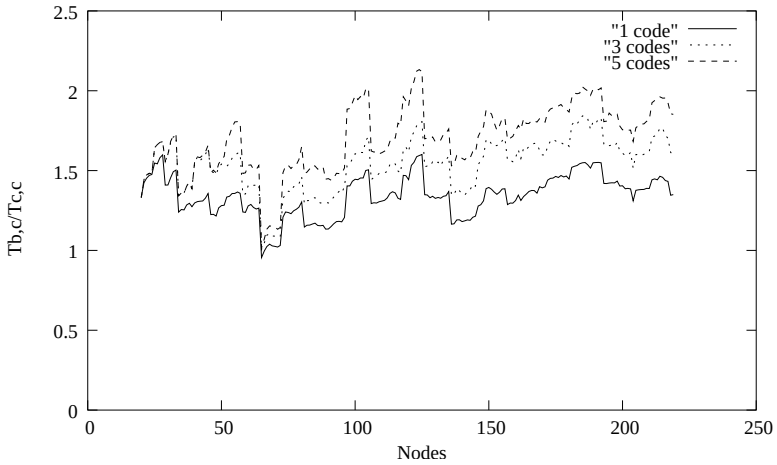

Figure 26. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.2 nodes/unit^2$**.**

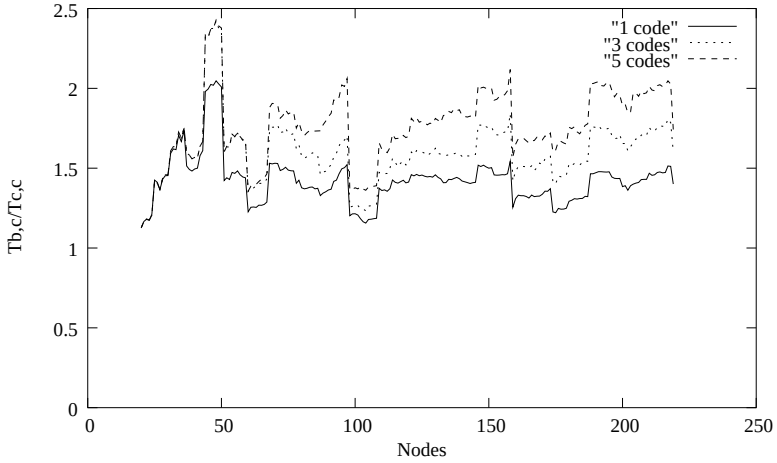

Figure 27. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.3 nodes/unit^2$.

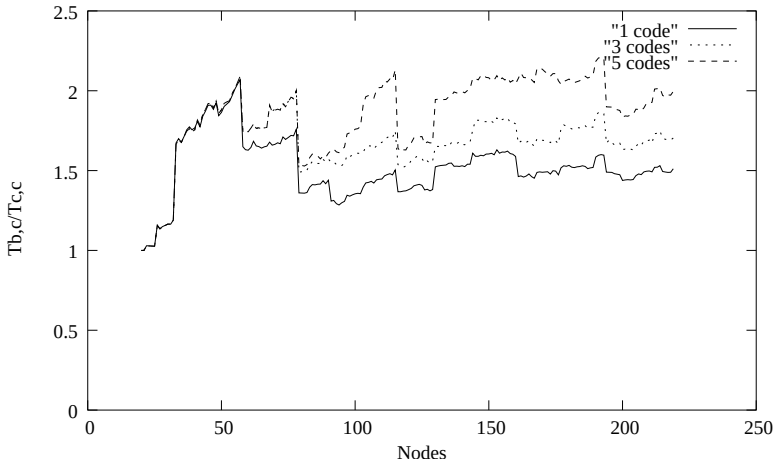

Figure 28. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.4 nodes/unit^2$.

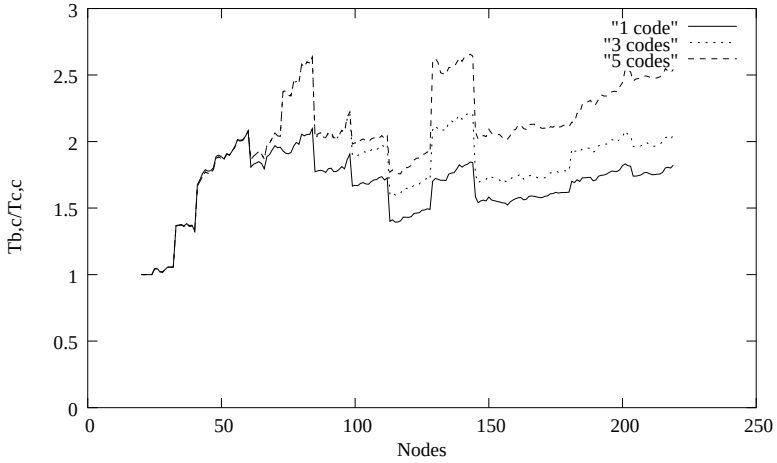

Figure 29. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.5 nodes/unit^2$.

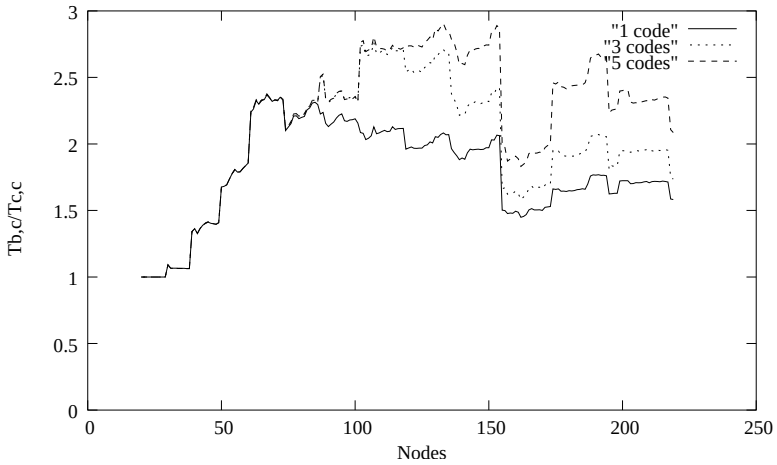

Figure 30. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.6 nodes/unit^2$.

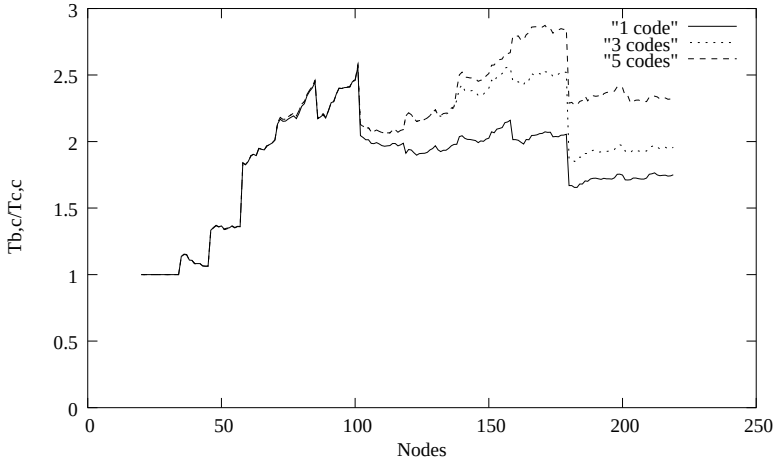

Figure 31. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.7 nodes/unit^2$.

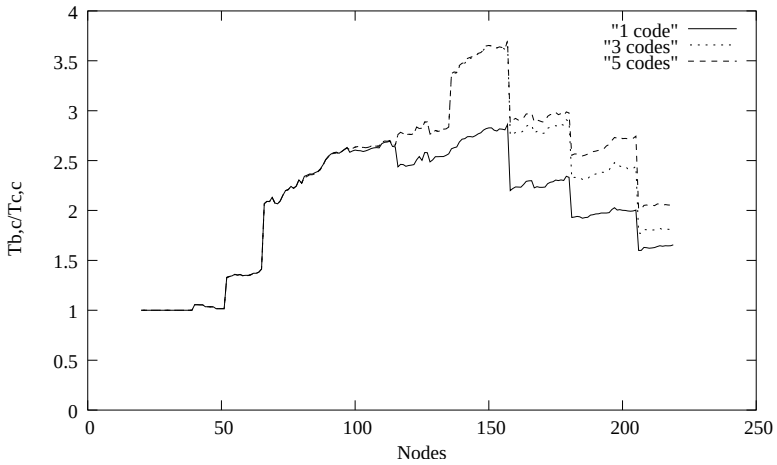

Figure 32. **The ratio of total duration for non-pipelined convergecasting over trees generated by broadcast approach of [3] and by CTCCAA algorithm respectively, for 1,3 and 5 codes on a network with node density** $0.8 nodes/unit^2$.

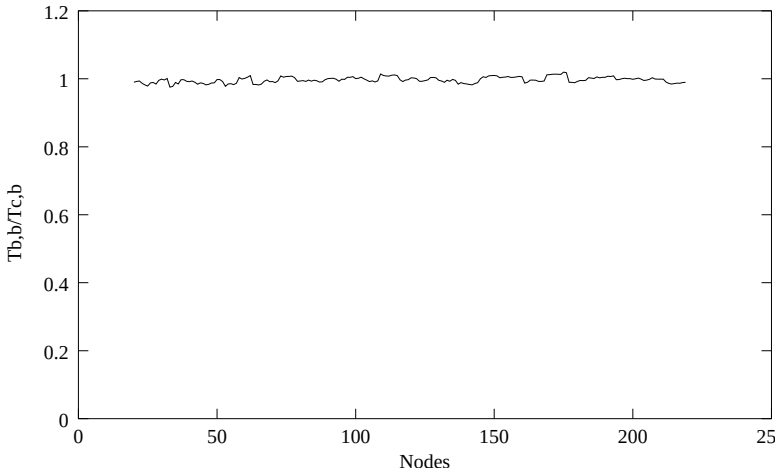

Figure 33. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.1 nodes/unit^2$.

The second version of the centralized algorithm of [3] takes as input this tree and allocates a valid schedule for broadcasting over this tree.

Simulations were carried out over randomly generated graphs with node densities varying from 0.1 $node/unit^2$ to 0.8 $nodes/unit^2$. We compared the total time required for broadcasting over this tree with the total duration required for broadcasting over a tree constructed by the broadcast algorithm of [3]. The total time required for broadcasting over the tree generated by CTCCAA algorithm ($T_{c,b}$) was compared with the total duration required for broadcasting over a tree constructed by the broadcast algorithm of [3] ($T_{b,b}$). From figure 33 to figure 40 shows the result obtained from simulations that were conducted. From the results we can observe that the tree constructed by CTCCAA works equally well for broadcasting too.

The results show that the CTCCAA constructs a tree and constructs a schedule in such a way that the latency for convergecasting (both pipelined and non-pipelined) is less. The results also show that the same tree is also good for broadcasting.

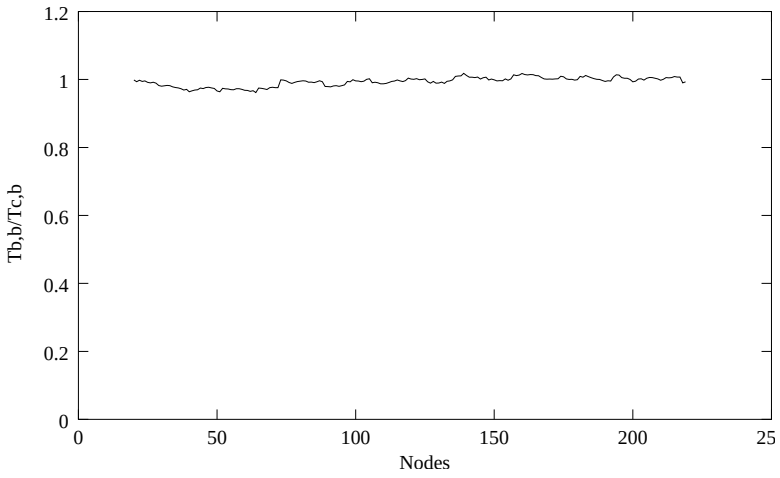

Figure 34. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.2 nodes/unit^2$**.**

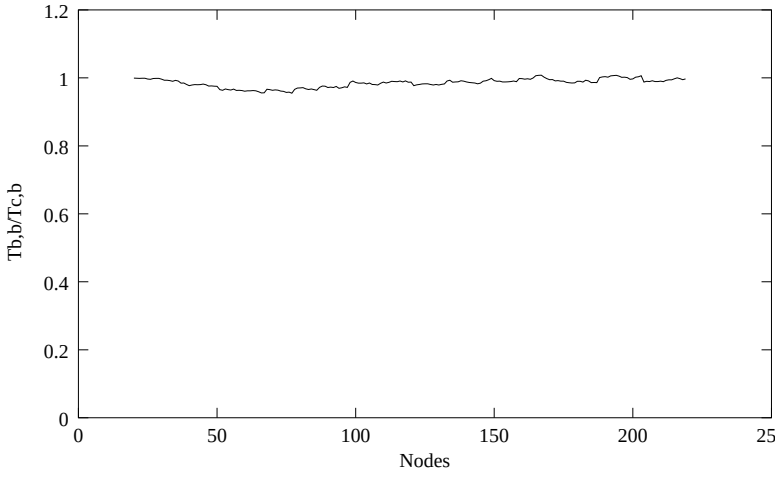

Figure 35. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.3 nodes/unit^2$**.**

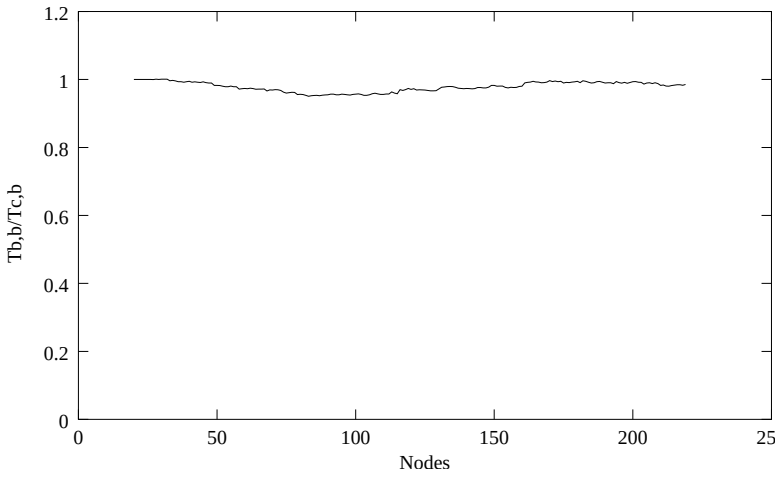

Figure 36. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.4 nodes/unit^2$**.**

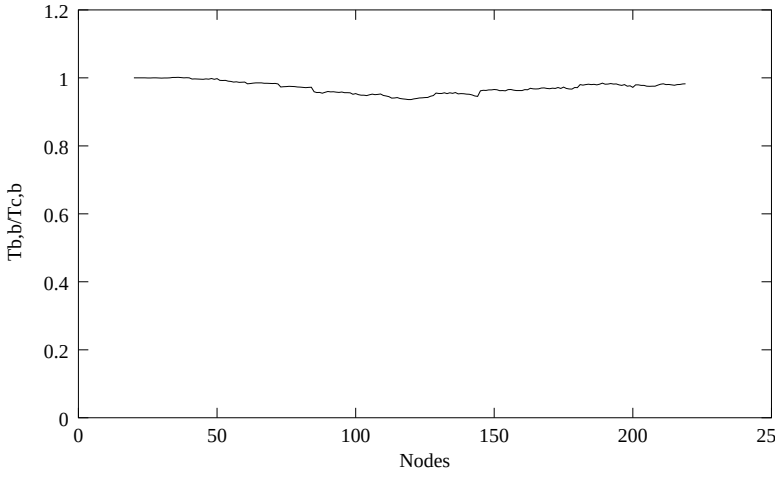

Figure 37. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.5 nodes/unit^2$**.**

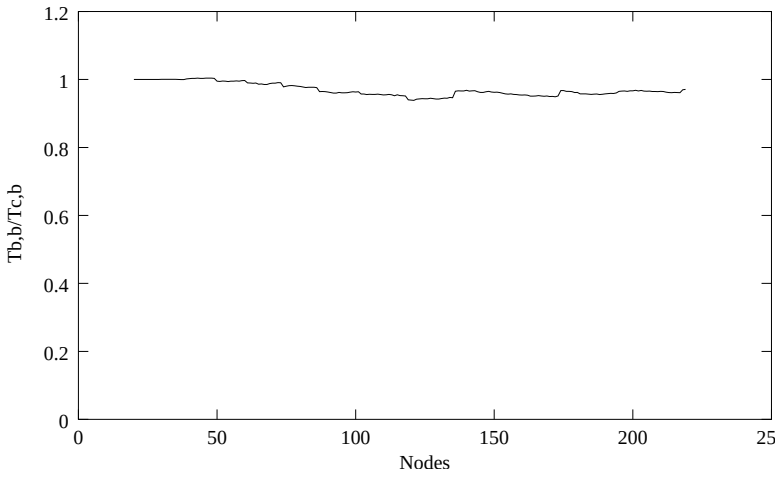

Figure 38. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.6 nodes/unit^2$.

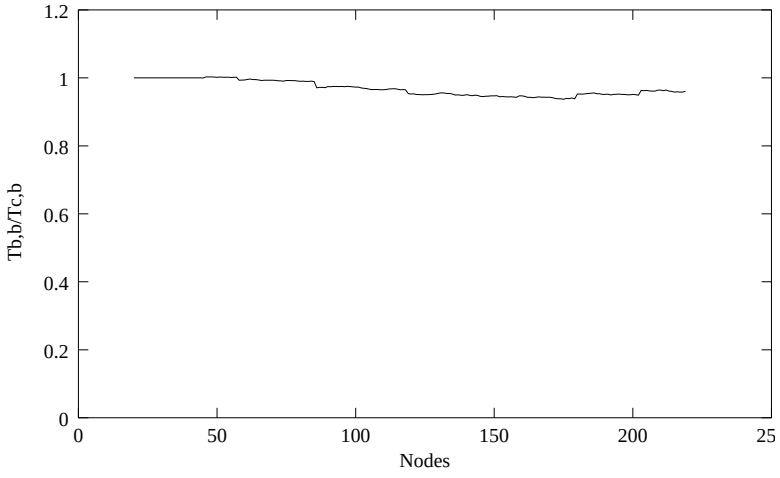

Figure 39. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.7 nodes/unit^2$.

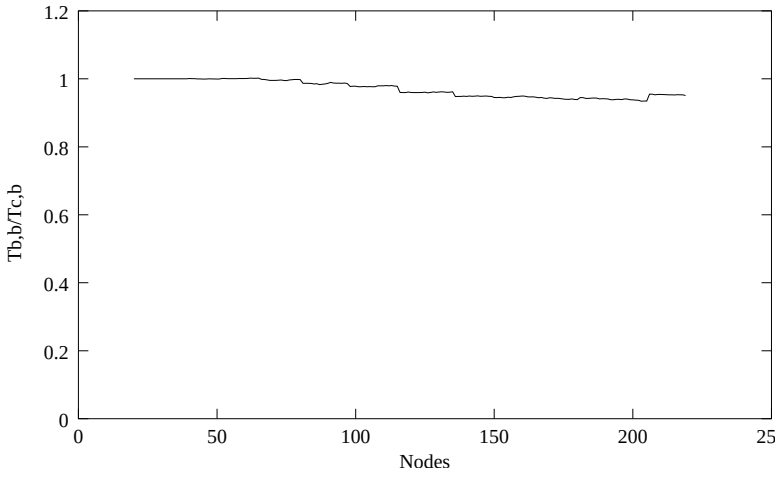

Figure 40. **The ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [3]for network with node density** $0.8 nodes/unit^2$.

CHAPTER 6

Conclusion

This thesis successfully proposed an algorithm that constructs a tree and assigns schedule for convergecasting. The results shows the need for a new tree convergecasting and the disadvantage of using a broadcast tree. CTCCAA algorithm constructs the tree such that it reduces the total time taken for both pipelined and non-pipelined convergecasting and this will be useful in reducing the wastage of power. This same tree can be used even for broadcasting and performs as efficient as a tree explicitly constructed for broadcasting.

CTCCAA allocates schedules that induces energy wastage by nodes, when the data they transmit is not equal to the slot size. The slot size is dependent upon the node that will transmit the maximum amount of data. A variation to this would be to allocates slot size based on the node that collects the smallest amount of data. For nodes that transmit data that is greater than this size the algorithm could allocate more than one slot that will aid the node in transmitting the complete data it had collected. This helps in avoiding the energy wastage in idle state.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci Wireless sensor networks: a survey In Computer Networks, Volume 38(2002) page(s) 393-422

[2] K. Bruce Jacobson Biosensors and Other Medical and Environmental Probes In http://www.ornl.gov/ORNLReview/rev29_3/text/biosens.htm

[3] I. Chlamtac and S. Kutten Tree-based Broadcasting in Multihop Radio Networks. In *IEEE Transactions on Computers, Volume C-36, No. 10, October 1987.*

[4] I. Chlamtac and O. Weinstein The Wave Expansion Approach to Broadcasting in Multihop Radio Networks. *IEEE Transactions on Communications, VOL. 39, NO. 3, March 1991.*

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest. *Introduction to Algorithms.*

[6] J. C. Culberson and F. Luo Exploring the k-colorable Landscape with Iterated Greedy. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 26, 1996.*

[7] A. Gamst Some Lower Bounds for a Class of Frequency Assignment Problems *IEEE Transactions on Vehicular Technology, VOL. VT-35, No. 1, Febraury 1986*

[8] H. Haas, S. McLaughlin and G. J. R. Povey A novel interference resolving algorithm for

TDD TD-CDMA mode in UMTS. In *Proceeding of International Symposium. Personal, Indoor and Mobile Radio Communications PIMRC 2000, U.K., London, U.K., Sept. 18-21, page(s) 1231-1335*

[9] H. Haas and S. McLaughin A dynamic channel assignment algorithm for a hybrid TDMA/CDMA-TDD interface using the novel ts-opposing technique In IEEE journal on selected areas in communications, VOL. 19, No. 10, October 2001 page(s) 1831-1846

[10] W. K. Hale Frequency Assignment: Theory and Applications. In *Proceedings Of the IEEE, Volume. 68, No. 12, December 1980.*

[11] L. Hu Distributed code assignment for CDMA packet radio networks In IEEE/ACM Transactions on networking, VOL. 1, No. 6, December 1993

[12] C. Intanagonwiwat, R. Govindan, D. Estrin Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks In*Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)*

[13] I. Katzela and M. Naghshineh. Channel assignment schemes for cellular mobile telecommunication systems. *IEEE Personal Communications, Volume 3:10–31, June 1996.*

[14] L. Kou, G. Markowsky and L. Berman A Fast Algorithm for Steiner Trees In *ACTA Informatica VOL. 15, 141-145 (1891)*

[15] B. Krishanamachari, D. Estrin and S. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks In *International Workshop of Distributed Event Based Systems (DEBS), Vienna, Austria, July 2002*

[16] S. Lindsey, C. Raghavendra and K. M. Sivalingam Data Gathering Algorithms in Sensor Networks Using Energy Metric In*IEEE Transactions on Parallel and Distributed Systems, VOL 13, NO. 9 September 2002*

[17] S. Ramanathan and E. L. Lloyd Scheduling Algorithms for Multihop Radio Networks In IEEE/ACM Transactions on Networking, VOL 1 page(s) 166-177, April 1993

[18] R. Ramaswami and K. K. Parhi Distributed Scheduling of Broadcasts in a Radio Network In INFOCOM, VOL. 2, page(s) 497-504, 1989

[19] L. Schwiebert, S. K. S. Gupta, and J. Weinmann *et al.* Research Challenges in Wireless Networks of Biomedical Sensors. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '01)*, July 2001.

[20] A. Sen and J. M. Capone Scheduling in Packet Radio Networks - A New Approach In *Proceedings of Global Telecommunications Conference, page(s) 650-654, 1999*

[21] Wave Wireless Networking Inc. Direct Sequence Vs. Frequency Hopping. In http://www.newwaveinstruments.com/resources/

[22] J. E. Wieselthier and A. Ephremides A new class of protocols for multiple access in Satellite Networks In IEEE Transactions on Automatic control, VOL. AC-25, No. 5, October 1980

[23] J. E. Wieselthier and A. Ephremides A distributed reservation scheme for spread spectrum multiple access channels In Proceedings of GLOBECOM, Sand Diego, CA, Nov 1983, page(s) 659-665