# Optimization of Collection Tree Protocol

By

**Harleen Kaur**

**10MCEC21**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AHMEDABAD-382481**

**May 2012**

# Optimization of Collection Tree Protocol

**Major Project**

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

**Harleen Kaur**

**(10MCEC21)**

Guided By

**Prof. Gaurang Raval**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AHMEDABAD-382481**

**May 2012**

# DECLARATION

I, **Harleen Kaur**, **10MCEC21**, give undertaking that the Major Project entitled **"Optimization of Collection Tree Protocol in Wireless Sensor Networks"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Institute of Technology of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

**Harleen Kaur**

# Certificate

This is to certify that the Major Project entitled "Optimization of Collection Tree Protocol" submitted by Harleen Kaur(10MCEC21), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

| | |
|---|---|
| Prof. Gaurang Raval | Dr.S.N.Pradhan |
| Guide, Associate Professor, | Professor and PG-Coordinator, |
| Department of C.S.E., | Department of C.S.E, |
| Institute of Technology, | Institute of Technology, |
| Nirma University, Ahmedabad. | Nirma University, Ahmedabad. |

| | |
|---|---|
| Prof.D.J.Patel | Dr.K.Kotecha |
| Professor and Head, | Director, |
| Department of C.S.E, | Institute of Technology, |
| Institute of Technology, | Nirma University, Ahmedabad. |
| Nirma University, Ahmedabad. | |

# Abstract

Collection tree protocol is a tree based protocol which is mainly used for aggregation. It is minimum cost routing tree which selects the path having minimum cost routing gradient. It will generate one or more trees to root which is called base station. When any node has data it sends data up the tree which is forwarded to root. CTP will provide link quality estimation among nearby neighbors. It will indicate number of transmissions from a node to send a packet to destination and whose acknowledgment is successfully received. It sends data to root by choosing a next hop. CTP is having a problem of fast energy depletion which is caused by congestion in the network. This problem can be diminished by several load balancing schemes to route information by other well connected routes. Also energy level of the network is not highly scalable. Whenever a node in a network is failed it is not able to do the function and it should not effect on overall network. CTP is not a fault tolerant network because if any node fails due to energy depletion, packets will drop. When more number of nodes are added in the network, congestion will take place and so it does not have an additional energy to choose an optimal path. In revised CTP broadcast fault tolerant algorithm using CTP has been implemented. To enhance the energy level of a network RSSI is used in broadcast fault tolerant algorithm. RSSI is used to determine amount of radio energy present in the radio channel. Once the network is cleared for sending the data by sender node, it will send its packet to destination and the packet is sent to destination hop by hop. Once the packet is reached to destination, it will measure its RSSI value and also the signal strength of wireless network at different power levels which will enhance the lifetime of the network. CTP suffers a problem by disseminating data statically in network. So in revised CTP line mobility model is integrated which will send data dynamically to sink node.

# Acknowledgements

My deepest thanks to **Prof. Gaurang Raval**, Associate Professor, Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad the Guide of the project that I undertook for giving him valuable inputs and correcting various documents of mine with attention and care. He has taken the pain to go through the project and make necessary amendments as and when needed.

My deep sense of gratitude to **Dr.S.N.Pradhan**, Professor and PG-Coordinator of Department of Computer Engineering, Institute of Technology, Nirma University, Ahmedabad for an exceptional support and continual encouragement throughout part one of the Major project.

I would like to thanks **Dr.Ketan Kotecha,** Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for his unmentionable support, providing basic infrastructure and healthy research environment.

I would also thank my Institution, all my faculty members in Department of Computer Science and my colleagues without whom this project would have been a distant reality. Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the first part of my dissertation work successfully.

**- Harleen Kaur**
**10MCEC21**

# Contents

# List of Figures

# Abbreviations

| | |
|---|---|
| CTP | Collection Tree Protocol |
| WSN | Wireless Sensor Network |
| ETX | Expected Transmission |
| ACK | Acknowledgement |
| THL | Time has Lived |
| RS-LPL | Relative synchronized low-power listening |
| RSSI | Received Signal Strength Indicator |
| SN | Sensor Network |
| BAN | Body Area Network |
| CPM | Closest Pattern Matching |

# Chapter 1

# Introduction

CTP's main task is to perform data collection in wireless sensor network. It contains data packets which is unicast. It is having two features reliability and efficiency. CTP should be able to send a packet greater than 90 percent. It should send a packet towards destination with minimum amount of transmission. This protocol will make one or more routing trees to the root which is called base station having minimum cost. CTP is having a routing problem which will lead to routing loop and hence will have network congestion. Due to network congestion packets will drop. Due to frequently changes in link quality it will have stale topology. It consists of two mechanisms which will deal with this problem. The first is data path validation which will send packet to route dynamically and detect the problem when it is not making progress towards destination. It uses data packet transmission and reception methodology. Second is adaptive beaconing which will send beacons faster when the topology is inconsistent due to which sender node which has send the data will hear it and it will adjust it route accordingly. It make use of Trickle algorithm which allows nodes to send very few control beacons when the topology is consistent and quickly detect the routing problem. It consist of routing metric called ETX(expected transmission). ETX of root is always 0 and ETX of node is defined as ETX of its parent plus ETX of link to its parent. The collection of ETX value is known as gradient value. CTP always chooses the path having minimum ETX value. If any node receives ETX value which is higher than its own then there will be a routing problem and eventually looses connectivity with a candidate parent.

CTP is a tree-based collection protocol. Some number of nodes in a network advertises themselves as tree roots. Nodes form a set of routing trees to these roots. CTP is address-

free in that a node does not send a packet to a particular root,instead, it implicitly chooses a root by choosing a next hop. Nodes generate routes to roots using a routing gradient. CTP assumes that it has link quality estimates of some nearby neighbors. These provide an estimate of the number of transmissions it takes for the node to send a unicast packet whose acknowledgment is successfully received.
How tree is constructed:
CTP is having ETX as routing gradient.First children enter into a network.Each parent will know about its children id below it. Parent is having higher gradient value then children.Each node in lower level will send packet to the node which is having lower ETX value which are in its communication range. Finally parent will send the packet up to the tree which are at intermediate level. And finally packets are routed to the sink node which is root having ETX value zero. All the packets are aggregated to the sink node.

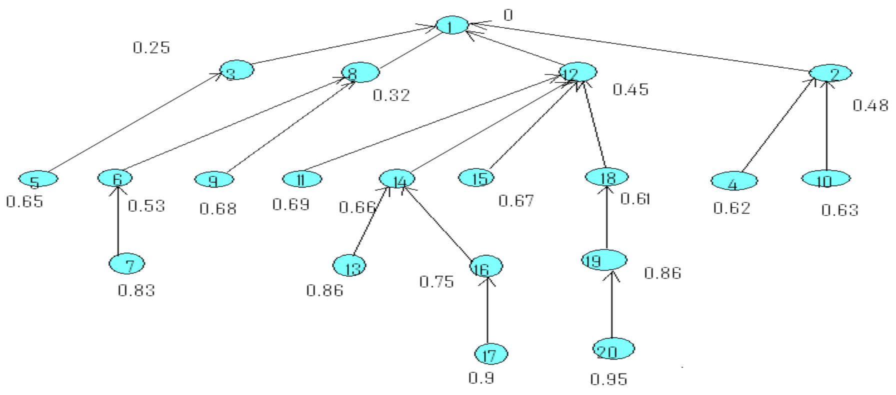


Figure 1.1: CTP tree construction

CTP uses expected transmissions ETX as its routing gradient. A root has an ETX of 0. The ETX of a node is defined as the ETX of its parent plus the ETX of its link to its parent. This additive measure assumes that nodes use link-level retransmissions. Given a choice of valid routes, CTP should choose the one with the lowest ETX value. CTP represents ETX values as 16-bit fixed-point real numbers with a precision of hundredths. Routing loops are a problem that can emerge in a CTP network.
CTP addresses loops through two mechanisms. First, every CTP packet contains a node's current gradient value. If CTP receives a data frame with a gradient value lower than its own, then this indicates that there is an inconsistency in the tree. CTP tries to resolve the inconsistency by broadcasting a beacon frame, with the hope that the node which

has sent the data frame will hear it and adjust its routes accordingly. If a collection of nodes is separated from the rest of the network, then they will form a loop whose ETX increases forever.

Packet duplication is an additional problem that can occur in CTP. Packet duplication occurs when a node receives a data frame successfully and transmits an ACK, but the ACK is not received. The sender retransmits the packet, and the receiver receives it a second time. This can have disastrous effects over multiple hops. For example, if each hop on average produces one duplicate, then on the first hop there will be two packets, on the second there will be four, on the third there will be eight, etc. Routing loop may cause a node to legitimately receive a packet more than once. CTP data frames have an additional time has lived (THL) field, which the routing layer increments on each hop [1].

## 1.1 Objective

The collection tree protocol is already available in TinyOS and Castalia.CTP is having a problem that it disseminates data statically.It does not consider RSSI, energy level of the network and individual node are also not highly scalable.Objective is to disseminate data dynamically to sink node and to include RSSI in CTP, which will enhance the energy level of the network.

## 1.2 Thesis Organization

The rest of the thesis is organized as follows.

**Chapter 1**: *Introduction*, This section gives an introduction of collection tree protocol.How it route data to nodes and problems in existing CTP.It describes about the objective of project and what are the goals to optimize collection tree protocol.

**Chapter 2**: *CTP Architecture*, This section includes the detailed architecture of collection tree protocol and explanation of three engines.

**Chapter 3**: *Structure of CTP*, This section includes the details of structure of CTP data frame.And describes about fields in CTP data frame.

**Chapter 4**: *Literature Survey*, This section includes the literature survey of collection tree protocol which describes about different methods about how data is routed to the sink and also the problem which occurs in CTP.

**Chapter 5**: *Applications*, This section describes about various applications of collection tree protocol.

**Chapter 6**: *Technologies used*, This section describes about tiny os and TOSSIM simulator. And how application is simulated in tiny os with the help of TOSSIM simulator.It also describes, how to configure a network with the help of python scripting language.It also describes about castalia wireless sensor network simulator which is based on OMNET++ platform.

**Chapter 7**: *Consistent Broadcast Algorithm*, This section describes about consistent broadcast algorithm and byzantine failure which occur in distributed system.

**Chapter 8**: *Broadcast Fault Tolerant Algorithm in Multi-Hop Network using CTP*, This section describes about an implementation of broadcast fault tolerant algorithm using CTP and describes about the method to send data to the base station.

**Chapter 9**: *Broadcast Fault Tolerant Algorithm in Multi-Hop Network using RSSI*, This section describes about an implementation of broadcast fault tolerant algorithm using RSSI and describes about the method to send data to the base station.

**Chapter 10**: *Implementation of CTP over Castalia*, This section describes about an implementation of CTP over castalia wireless sensor network simulator.And also describes about how line mobility model is integrated in CTP to send data dynamically and compare the results of existing and modified CTP.

**Chapter 11**: *Conclusion*

# Chapter 2

# CTP Architecture

CTP Architecture consist of three main logical software components:

- Link estimator
- Routing engine
- Forwarding engine

Link Estimator: is a module which is used to estimate the link of node using routing metric ETX. ETX of root is zero and ETX of node is defined as ETX of parent plus ETX of link to its parent. Node always selects the path having minimum ETX value. Collection of ETX value is known as gradient. If a node receives a packet from a node having lower gradient value then its own, then there will be routing inconsistency[2].

Routing Engine: is a module which routes packet in the tree towards sink and maintains routing table containing list of neighbors from which node can select its parent. The information in the table is filled with the help of beacons which has been transmitted. A node having an ETX and its value equal to n will be able to send data to sink with a total of n transmission. The rate at which beacons are sent are adjusted by trickle algorithm. The node will select its parent which is having lower ETX value. Each time a node receives any updation from its neighbor, it records its information in its routing table. Routing table has two uses: first one is to select the parent and to select the minimum routing path. Routing engine has two functions: UpdateRouteTask in which parent is selected rapidly before a node will send its own beacon. SendBeaconTask broadcast the current route information to others and also takes care of sending and receiving beacons[2].

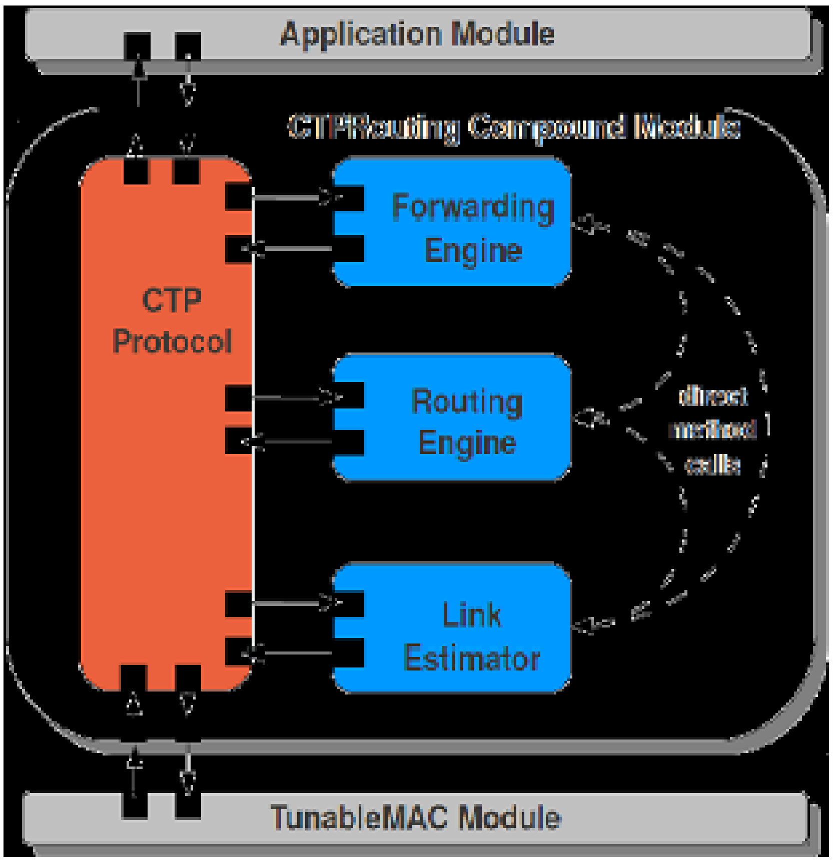


Figure 2.1: CTP Architecture from reference[2]

The neighbor table keeps the best candidates for being parents i.e., the nodes with the best path metric. The neighbor table does not store the full path metric.Table stores the parent's path metric, and the link quality to the parent is only added when the information is needed: (i) when choosing a parent and (ii) when choosing a route.

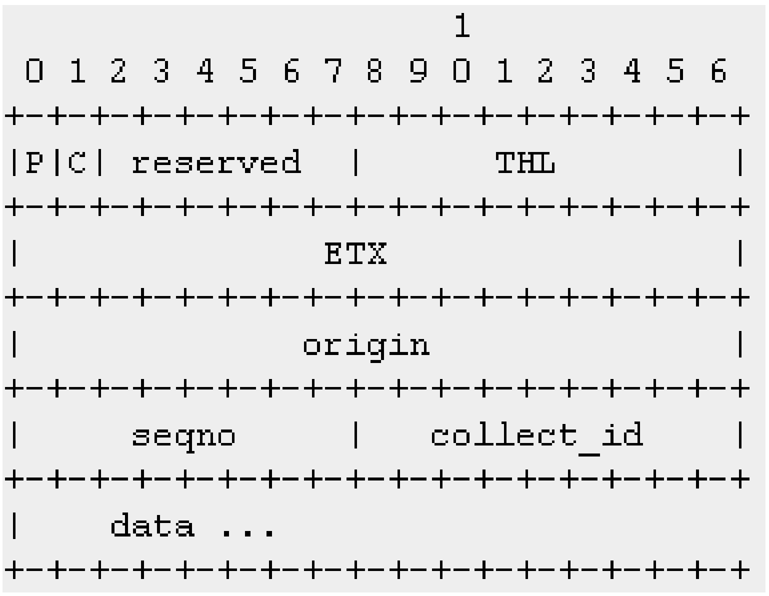
Routing table serves to define the node's own path metric, hopcount, and the set of child-parent links. In a sense the tree is defined to form a coherent propagation substrate for the path metrics.

Forwarding Engine: It forwards data packets to node and then ultimately it is forwarded towards root. Node will select the parent from routing table and will check the identifier of a current node and if the ETX of sender is greater than ETX of receiver than there will be routing inconsistency. For routing inconsistency data path validation and adaptive beaconing method is used[2].

# Chapter 3

# Structure of CTP

CTP Data Frame[2]:



```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P|C| reserved  |      THL         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ETX                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             origin               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     seqno     |   collect_id     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3.1: CTP data frame from reference [2]

- P(Routing Pull):- This bit allows nodes to request information from other nodes. If any node hears a packet with the P bit set, it should transmit a routing frame in the near future.

- C(Congestion Notification):- If a node transmits a CTP data frame it must set with the C bit field on the next data frame.

- THL(Time Has Lived):- When a node generates a data frame it sets THL to 0.When a node receives a CTP data frame, it must increment the THL value and if a node receives a THL of 255, it increments to 0.

- Origin:- The originating address of the packet. A node forwarding a data frame must not modify the origin field.
- Seqno(Sequence Number):- This field signifies the origin sequence number. The originating node sets this field, and a node forwarding a data frame must not modify it.
- Collect_id:- This field signifies the higher-level protocol identifier. The sender sets this field, and a node forwarding a data frame must not modify it.
- Data:- This field signifies the data payload, of zero or more bytes. A node forwarding a data frame must not modify the data payload.

CTP Routing Frame[2]:

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P|C| reserved  |      parent     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     parent    |       ETX       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ETX      |
+-+-+-+-+-+-+-+-+
```

Figure 3.2: CTP routing frame from reference [2]

- P(Routing Pull):- Same as data frame.
- C(Congestion Notification):- If a node drops a CTP data frame, it must set with the C field on the next routing frame it transmits.
- Parent:- The node's current parent.
- Metric:- The node's current routing metric value.

# Chapter 4

# Literature Survey

In[3], The author describes about an improved collection of tree protocol. This protocol implements the load balance into the collection tree protocol(CTP). However it is having an additional problem of not selecting an optimal path. CTP is having an problem of network jitter. It gives basic function of CTP protocol.

CTP has proved to be far better a data collection protocol, however, CTP only consider the most optimal path selection. It mainly cause two main problems: first, it causes a larger network jitter, that means in a time when routed to the congestion, the network will choose another path to continue to send again until the new path congestion, so the network is formed jitter. Second is that node in the optimal path causing energy loss which is too large, not only cause the entire network of energy imbalance, but also may reduce the energy lifetime of the entire network and third problem is that although the CTP's ETX value can detect congestion, and choose another path, but inevitably encounter congestion, which may reduce the efficiency of network data collection.

Normally, with the increase of routing efficiency, the energy consumption will increase as well. However, ICTP keeps the reliability and the efficiency of the network while the energy consumption does not increase. The reason was mainly due to the route of the packets which is generated by the probability in ICTP. However, since ICTP makes use of both long but good link quality path and short but bad link quality path, the reliability may drop a little. On the other hand the improvement of congestion will increase the reliability. In terms of energy balance: the choice of path based on probability may end with a bad link quality one and leads a node to resend the data many times in order to complete a single-hop data transmission which results in additional energy consumption.

However, considering the probability model based on energy in ICTP, the path chosen by the algorithm always contains the largest statistic energy and ensures the balance of energy consumption. At the same time, congestion factor is taken into account in ICTP and the result is that the possibility of congestion reduces.
In order to solve these problems, the best way is to use the probability of selection, which is determined by the merits of the path and the size of the energy. This may avoid the congestion caused by excessive load nodes which causes network jitter and the problem of declining. However, this may introduce a new problem that if the choice of the path is not the optimal path, it may result in packet loss rate increasing and additional single-hop energy consumption.ICTP works in following manner:

- It works by the design of routing table.
- It use link estimates same as candidate nodes.
- It uses probability to determine the parent node.

It defines statistical method about how to enhance the speed from source to target node. It uses the average number of hops to measure the efficiency of routing protocol simulation and uses three kinds of topology linear, mesh and random. In case of linear topology and mesh topology the node is only 2m to others and its link quality estimation is poor as compared to random topology.
In[1],This paper tells about how to implement byzantine fault tolerant algorithm in multi hop wireless sensor network with the help of collection tree protocol. It discuss about byzantine algorithm which assumes that network is an n mote connected and there is only one input state and the goal is for all the motes to eventually output decisions from the set V. Every known technique for building systems that resist byzantine faults has at least one of the following weaknesses:- its throughput does not increase with scale, it addresses only a narrow academic problem, it does not support computation and does not address general byzantine faults. It implements a consistent broadcast protocol for implementing byzantine fault tolerant algorithms. There are three conditions which validates algorithm, all correct processes eventually reach a decision regarding the value order they have been given. All processes must decide on the same value v. If the source process is a correct process, all processes have to decide on the value that was original given by the source process. It gives about description that CTP on data link layer has following 4

problems: Its throughput does not increase with scale, it does not support computation and it does not address general byzantine faults. It addresses only a narrow academic problem. It gives about description of how basic CTP works. It gives about description of three dissemination protocol drip, div and dhv protocol. And comparison between three protocols which shows that drip uses more power than dip and dhv. The algorithm which is implemented on multi-hop networks detects faulty motes and is scalable upto large number of motes.

In [4], the author describes about the RS-LPL-relative synchronized low-power listening which uses the relative time synchronization to improve the traditional low power listening scheme, and realize the RS-LPL in Collection Tree Protocol to enhance the network energy efficiency and prolong the network lifetime. The experiment results in a network consisted of mote shows that RS-LPL is more energy efficient than traditional low power listening.

The paper focus on the decreasing node energy consumption approach and propose relatively synchronous low power listening (RS-LPL). Compared with traditional LPL, RS-LPL introduces a relative time synchronization model for predicting wake-up time, which eliminates long preamble, and enhance the energy efficiency of node. Meanwhile, combining with the CTP, it achieves RS-LPL in the Telosb platform, and experiment results show that the RS-LPL is more energy efficient than traditional LPL and prolongs the network lifetime. The relative time synchronization model and transmission model is proposed.

In relative time synchronization model, nodes maintain their own local clock independently and store the clock offset and clock skew of neighbour nodes. According to this information, node establishes the relations with neighbour's clock and achieves mutual conversion between nodes clock for relative time synchronization.

In transmission model, the process of establishing and maintaining network topology, CTP will broadcast a series of packets. By embedding SYNC field into these broadcast packets, node can establish a synchronous neighbour table, which contain some information about synchronization points. According to the content of synchronous neighbour table, node can transmit data packets with minimum preamble so that to save energy.

In [5],the author describes about SenZip architecture in wireless sensor network.In wireless sensor network, compression is essential for enhancing the lifetime of the network.

This progress has not been made in distributed system. To fill the gap SenZip architecture is introduced, which will interact with the standard networking components. This architecture is designed for en-route compression. This compression will happen en-route to the sink. For efficiency and scalability a communication is needed which will determine which node will perform compression over what data and how that data is routed. It is an architecture that will be deployed on motes which can achieve configuration and modularity. This will encompass different compression technique which will interact with standard networking components. Since compression will be done in a distributed manner, the computation will not be held on central node but it must be carried out across multiple nodes. Once the data from a given node is compressed it will be routed to the sink along a routing tree like CTP. The primary goal of SenZip architecture is modularity, distributed configuration and reconfiguration.
It consists of two modules:

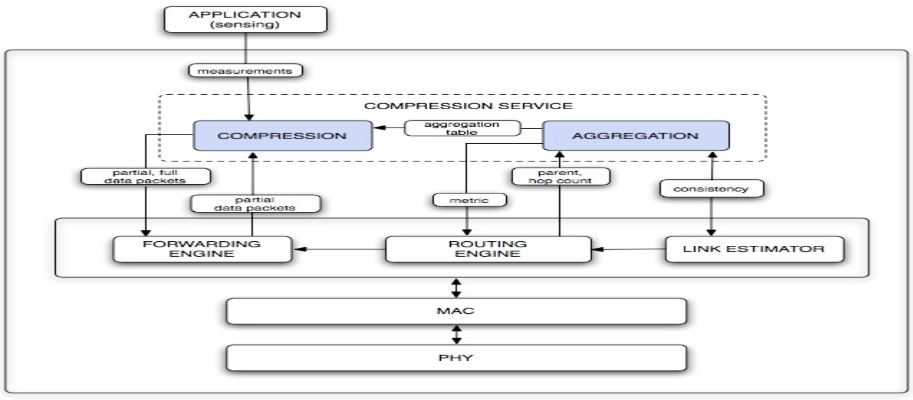


Figure 4.1: SenZip Architecture from reference[5]

- Aggregation module: It will disseminates data and gather information for maintaining the aggregation tree by exchanging messages. This information will be added in aggregation table. The aggregation graph allows the definition of generic table that will work for different compression scheme.The routing component signals changes in parent in routing tree.At this point, the aggregation component sends an ADD beacon to the new parent and a DELETE beacon to the old parent.The old and new parents update their aggregation tables accordingly.

- Compression module: This module describes the role played by the node which computations to perform and for which nodes, the parameters involved in computation and ordering information - the order in which nodes process and forward data.It receives raw measurements from the application and packets with data that needs further processing from forwarding. This module performs further processing over the partially processed data in storage and initiates processing for data of the node itself. Data that is still partially processed is packetized and sent to forwarding. For data that is fully processed, it checks if enough has been buffered in storage to fill a packet. If yes, performs quantization and bit reduction operations, and sends the packet to forwarding.When the aggregation component signals changes in the aggregation table,the compression component allocates/de-allocates memory for storing the data of children in aggregation tree.

Networking Components

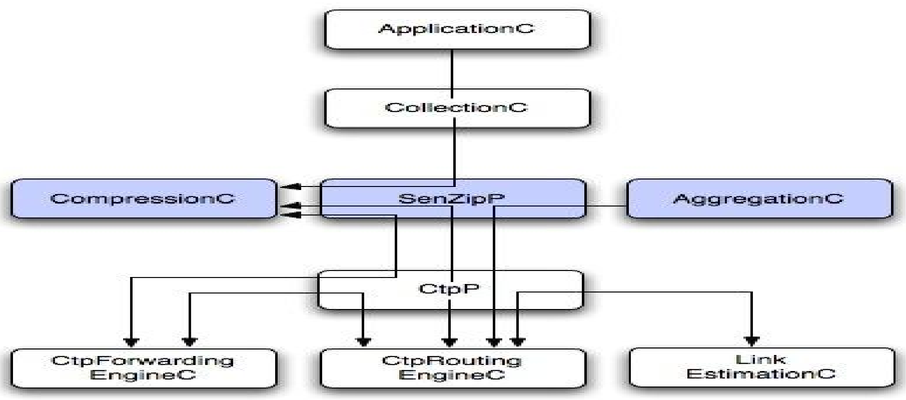


Figure 4.2: SenZip over CTP from reference[5]

- Routing engine: In addition to the standard routing functionality, this component in SenZip has an extra interface to the compression service. It reports information of path routing that is relevant for the local aggregation. Optionally decisions on changing parent can be coordinated with the compression service, which can also provide a specific metric for the routing cost.
- Forwarding engine: While partially processed data from nodes in the local aggregation tree is allowed to be intercepted by the compression service, fully processed data is forwarded directly along the route to the sink.
- Link estimator: Efficient link estimation requires a limited choice of links to monitor. To remove a link that is part of the current aggregation tree, joint decision

has to be made with the compression service to maintain consistency in the data processing.

How System Works

- Routing tree: When nodes are switched on the CTP routing algorithm builds a tree rooted at the sink.Each node chooses a parent in the tree for which the ETX metric is minimized.
- Local aggregation tree: Once the parent is known,beacons are exchanged to build local aggregation tree.When nodes select a parent,they send it a beacon for inclusion in the parent's aggregation tree.
- Start gathering: When a START message is sent to any node in the network,it broad- casts this message and initiates sensor measurements.When other nodes receive this START message,they re-broadcast it and initiate sensing.
- Compression: Once started,nodes collect sensor measurements in packets and send to the parent in tree. The parent applies the DPCM transform for each child and generates coeffcients.These coeffcients are passed through a fixed quantization encoder for compression. When there are enough bits from compressed coeffcients to fill a packet, they are palletized and send to the sink.
- Reconfiguration: When changes in the routing tree and hence in local aggregation tree occur,beacons are exchanged locally to update this information.

# Chapter 5

# Applications

Most sensor networks are used to collect information from the physical world.Sensor networks deployed to monitor micro-climates in agriculture farms and deployments that measure energy consumption in office or residential buildings. The nodes in these networks collect information about the physical world using their sensors and relay the sensor readings to a central base station or server using multi-hop wireless communication.
In the application domain of battlefield surveillance, environmental monitoring systems, and homeland security control, wireless sensor networks are deployed to collect and aggregate sensing data from scattered field sensors, and send to the sink node for further processing. For such a type of network, the hierarchical architecture is widely used in which the distributed sensor nodes are grouped into various segments, namely clusters, with each cluster has a head.

# Chapter 6

# Technologies Used

To study how basic CTP works, it needs to implement a basic collection tree protocol in tiny os. First of all we have installed tiny OS 2.1.1 on machine which is open source operating system designed for wireless sensor network. It is a light weight operating system specifically designed for low power wireless sensors. It consist of TOSSIM simulator which simulates entire tiny OS applications. It is a discrete event simulator. When it runs, it pulls events of the event queue (sorted by time) and executes them.

At a high level, TinyOS provides three things to make writing systems and applications easier:

- A component model, which defines how you write small, reusable pieces of code and compose them into larger abstractions.
- A concurrent execution model, which defines how components interleave their computations as well as how interrupt and non-interrupt code interact.
- Application programming interfaces (APIs), services, component libraries and an overall component structure that simplify writing new applications and services.The component model is grounded in nesC. It allows you to write pieces of reusable code which explicitly declare their dependencies.

## 6.1 Execution Model

Compiling TinyOS applications produces a single binary image that assumes it has complete control of the hardware. Therefore, a mote only runs one TinyOS image at a time.

An image consists of the components needed for a single application. As most mote platforms do not have hardware-based memory protection, there is no separation between a "user" address space and a "system" address space, there is only one address space that all components share. This is why many TinyOS components try to keep their state private and avoid passing pointers: since there is no hardware protection, the best way to keep memory uncorrupted is to share it as little as possible.

## 6.2 Components and Interfaces

A nesC application consists of one or more components assembled, or wired, to form an application executable. Components define two scopes: one for their specification which contains the names of their interfaces, and a second scope for their implementation. A component provides and uses interfaces. The provided interfaces are intended to represent the functionality that the component provides to its user in its specification, the used interfaces represent the functionality the component needs to perform its job in its implementation.

Interfaces are bidirectional: they specify a set of commands, which are functions to be implemented by the interface's provider, and a set of events, which are functions to be implemented by the interface's user. For a component to call the commands in an interface, it must implement the events of that interface. A single component may use or provide multiple interfaces and multiple instances of the same interface. The set of interfaces which a component provides together with the set of interfaces that a component uses is considered that component's signature.

## 6.3 Configurations and Modules

There are two types of components in nesC: modules and configurations. Modules provide the implementations of one or more interfaces. Configurations are used to assemble other components together, connecting interfaces used by components to interfaces provided by others. Every nesC application is described by a top-level configuration that wires together the components inside.

## 6.4 TOSSIM with Python

Python is an interpreted, general-purpose high-level programming language whose design philosophy emphasizes code readability. Python aims to combine "remarkable power with very clear syntax", and its standard library is large and comprehensive. Its use of indentation for block delimiters is unique among popular programming languages.In TOSSIM, we can either write a script to tell Python to run it.

We will start with Python in interactive mode first, though. To start the Python interpreter, type: $python. The version may be different, based on your installation. The first thing we need to do is import TOSSIM and create a TOSSIM object which can be done by writing

$fromTOSSIMimport*$

$t = Tossim([])$

The way we run a TOSSIM simulation is with the runNextEvent function $t.runNextEvent()$. When we tell TOSSIM to run the next event, it returns 0. This means that there was no next event to run. In our case, there's no next event because we haven't told any nodes to boot. This snippet of code will tell mote 32 to boot at time 45654 (in simulation ticks) and run its first event (booting):

$m = t.getNode(32)$

$m.bootAtTime(45654)$

$t.runNextEvent()$

## 6.5 Configuring a Network

When we start TOSSIM, no node can communicate with any other. In order to be able to simulate network behaviour, we have to specify a network topology. Internally, TOSSIM is structured so that we can easily change the underlying radio simulation. The default TOSSIM radio model is signal-strength based. We provide a set of data to the simulator that describes the propagation strengths. We also specify noise floor, and receiver sensitivity.We can control the radio simulation through a Python radio object:

$fromTOSSIMimport*$

$t = Tossim([])$

$r = t.radio()$

In addition to the radio propagation model above, TOSSIM also simulates the RF noise and interference a node hears, both from other nodes as well as outside sources. It uses the Closest Pattern Matching (CPM) algorithm. CPM takes a noise trace as input and generates a statistical model from it. This model can capture bursts of interference and other correlated phenomena, such that it greatly improves the quality of the RF simulation. To configure CPM, it needs to feed a noise trace. This can be accomplished by calling "addNoiseTraceReading" on a Mote object. Once the entire noise is traced, it calls "createNoiseModel" on the node. PowerTOSSIM-Z is a power modeling extension to TOSSIM for TinyOS, version 2.x. For compilation,command is make micaz sim. After that it simulates the entire tiny os application. There will be a post process file which is used as trace file and then it extracts data file from post process file. The design models power consumption for the Micaz sensor mote. It helps in analyzing power model and creates report in .csv files.

## 6.6 Castalia Simulator using OMNET++ Platform

Castalia is a simulator for Wireless Sensor Networks (WSN), Body Area Networks (BAN) and generally networks of low-power embedded devices. It is based on the OMNET++ platform.Castalia can also be used to evaluate different platform characteristics for specific applications, since it is highly parametric, and can simulate a wide range of platforms. The main features of Castalia are[15]:

Advanced channel model based on empirically measured data.

- Model defines a map of path loss, not simply connections between nodes.
- Complex model for temporal variation of path loss.
- Probability of reception based on SINR, packet size, modulation type. PSK FSK supported, custom modulation allowed by defining SNR-BER curve.
- Highly flexible physical process model.
- Sensing device noise, bias, and power consumption.
- MAC and routing protocols available.
- Designed for adaptation and expansion.

- Node clock drift.
- Extended sensing modelling provisions.
- Multiple TX power levels with individual node variations allowed.
- States with different power consumption and delays switching between them.

OMNET's basic concepts are modules and messages. A simple module is the basic unit of execution. It accepts messages from other modules or itself, and according to the message, it executes a piece of code. The code can keep state that is altered when messages are received and can send new messages. There are also composite modules. A composite module is just a construction of simple and/or other composite modules.

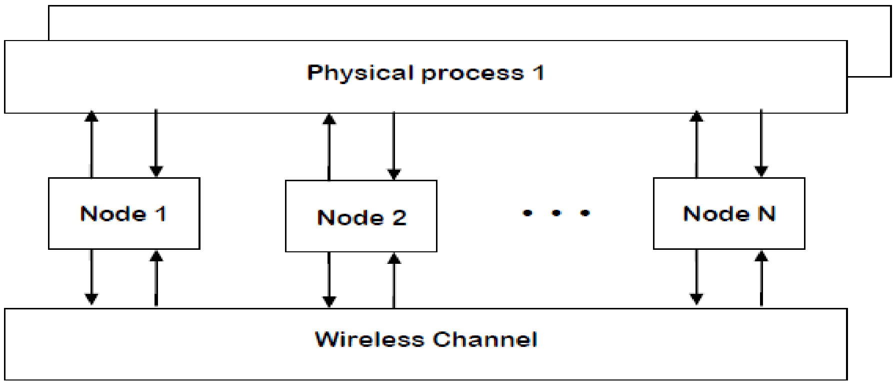


Figure 6.1: The modules and their connections in Castalia from reference[15]

In this figure nodes do not connect to each other directly but through the wireless channel modules. The arrows signify message passing from one module to another. When a node has a packet to send this goes to the wireless channel which then decides which nodes should receive the packet. The nodes are also linked through the physical processes that they monitor. The nodes sample the physical process in space and time by sending a message to the corresponding module, to get their sensor readings. There can be multiple physical processes, representing the multiple sensing devices that a node has. The node module is a composite one. Implementation in Castalia is done with the use of the OMNET++ NED language. With this language we can easily define modules, i.e., define a module name, module parameters, and module interface (gates in and gates out) and possible submodule structure.Files with the suffix ".ned" contain NED language code. The Castalia structure is also reflected in the hierarchy of directories in the source

code. Every module corresponds to a directory which always contains a .ned file that defines the module. If the module is composite then there are subdirectories to define the submodules. If it is a simple module then there is C++ code (.cc, .h files) to define its behavior. This complete hierarchy of .ned files defines the overall structure of the Castalia simulator.
Castalia has a modular structure with many interconnecting modules. Each one of the modules has one or more parameters that affect its behavior. An NED file (file with extension .ned in the Castalia source code) defines the basic structure of a module by defining its input/output gates and its parameters. In the NED file we can also define default values for the parameters. A configuration file (usually named omnetpp.ini and residing in the Simulations dir tree) assigns values to parameters, or just reassigns them to a different value from their default one. This way we can build a great variety of simulation scenarios. Open the file Simulations/radioTest/omnetpp.ini.In this file parameters such as the simulation time,number of nodes and field size values must be defined in this section.

```
[General] include ../Parameters/Castalia.ini
sim-time-limit = 100s
SN.field_x = 200 # meters
SN.field_y = 200 # meters
SN.numNodes = 3
```

SN is the topmost composite module. The name SN stands for Sensor Network.The parameter SN.deployment is a string that describes where the nodes are placed on the field.

When defined, SN.deployment can be one of the following types:

- uniform: nodes are placed in the field using a random uniform distribution.
- NxM: N and M are integer numbers. Nodes placed in a grid of N nodes by M nodes.
- NxMxK: same as above but for 3 dimensions.

Below two lines set 2 parameters of the Radio module. The RadioParametersFile is a specially formatted file defining the basic operational properties of a radio. The second parameter, TxOutputPower sets the power that the radio transmits its packets. We can access all the radio modules in all the nodes with the use of [*].

- $SN.node[*].Communication.Radio.RadioParametersFile = "Radio/CC2420.txt"$
- $SN.node[*].Communication.Radio.TxOutputPower = " - 5dBm"$

# Chapter 7

# Consistent Broadcast Algorithm

Distributed system generally having the problem of byzantine failure.The term byzantine was first used for the type of failure where faulty motes may exhibit completely unconstrained behavior. The Problem assumes that the network is an n-mote connected undirected graph with motes 1. . . n, where each mote knows the entire graph. Each mote starts with an input from a fixed value set V in a designated state component. For each mote, there is exactly one start state containing each input value. The goal is for the motes to eventually output decisions from the set V, by setting special decision state components to values in V with the possibility that a limited number (at most f) of motes might fail.
The objective of this project is to provide network designers a new algorithm that would aid in creating failure tolerant networks and hence create more robust networks that are reliable even when some motes do not function as desired.
Broadcast fault tolerant algorithm can be helpful to solve the byzantine failure problem which generally occurs in the distributed systems.This algorithm once implemented can have its use in detecting failure in real world applications like Smart Home, detection of forest fire, sewage system, and Smart environments or in fact at all the places where distributed networks are used for sensing.

This is a concept of consistent broadcast algorithm for solving Byzantine agreement for a general value set V.This mechanism is a way of ensuring a certain amount of coherence among the messages received by different motes.Here a mote i can broadcast a message of the form (m, i, r) at round r, and the message can be accepted by any of the motes (including i itself) at any subsequent round.

The consistent broadcast mechanism is required to satisfy the following three conditions:

- If non-faulty mote i broadcasts message (m, i, r) in round r, then the message is accepted by all non-faulty motes by round r + 1 (i.e., it is either accepted at round r or round r + 1).
- If non-faulty mote i does not broadcast message (m, i, r) in round r, then (m, i, r) is never accepted by any non-faulty mote.
- If any message (m, i, r) is accepted by any non-faulty mote j, say at round r, then it is accepted by all non-faulty motes by round r + 1.

The first condition says that non-faulty motes broadcasts are accepted quickly, while the second says that no messages are ever falsely attributed to non-faulty motes. The third condition says that any message that is accepted by a non-faulty mote (whether from a faulty or non-faulty sender) must also be accepted by every other non-faulty mote soon thereafter [1].

# Chapter 8

# Broadcast Fault Tolerant Algorithm in Multi-Hop Network using CTP

The algorithm is divided in different stages:

- First Stage: All nodes send HELO packet to base station using CTP and base station finds total number of nodes in network as shown below.

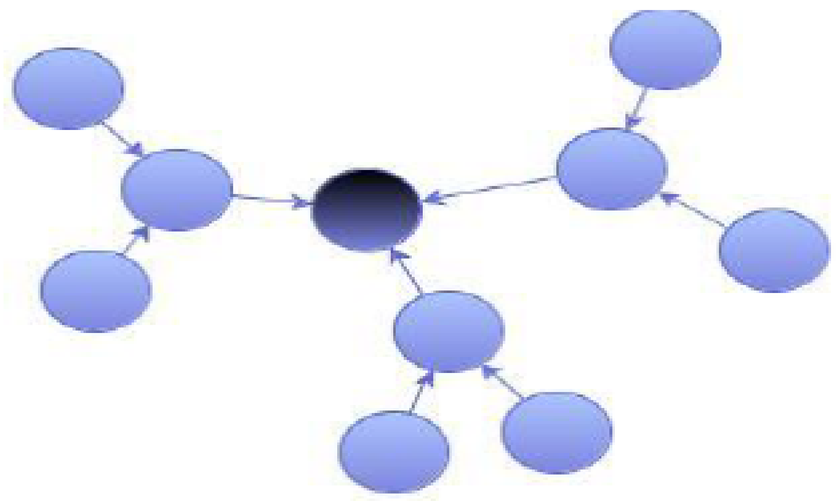


Figure 8.1: Nodes sending HELO packet to the base station using CTP from reference [1]

- Second Stage: Base station sends "n (number of nodes)" to all nodes using Dissemination as shown below.

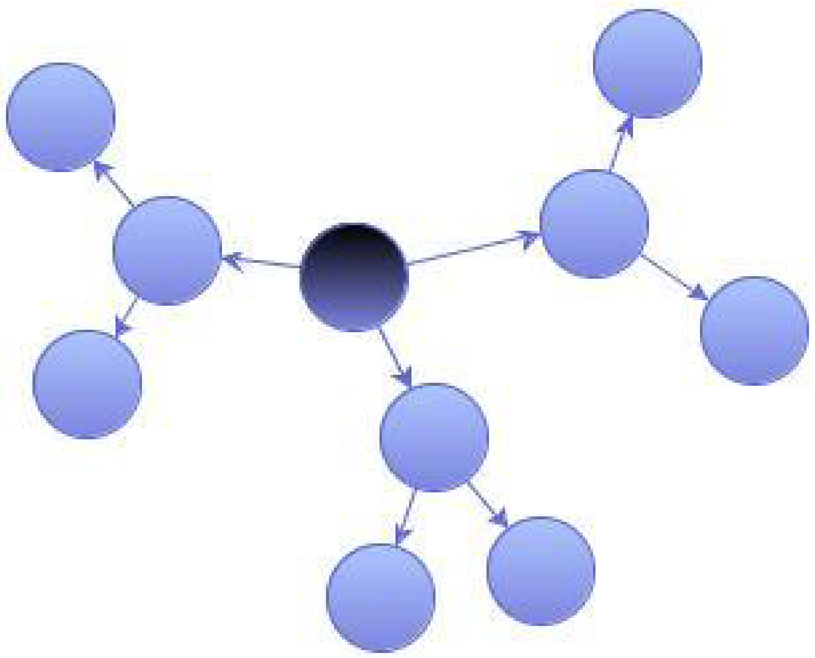


Figure 8.2: Base station sending information about number of nodes in the network from reference [1]

- Third Stage: In this stage one node will be randomly selected and will disseminate INIT packet in the network as shown below.

- Fourth Stage: After receiving INIT packet, nodes send ECHO packet. Nodes keep counter of ECHO packets.

- Fifth Stage: If any node has received n-f ECHO packets in previous round then it establishes successful communication.

Packet structures

- HELO Packet: The HELO packet is a 16bit packet that contains the node id of the node that is transmitting the packet. Through this the base station or the root of the tree in the Collection Tree Protocol receives the node id of the node that sent it the HELO packet.

- INIT Packet: This is a value that is disseminated over the network using the Drip protocol. The value in this case would be a 0. If a node receives a disseminated value of 0 then it behaves as having received an INIT packet.

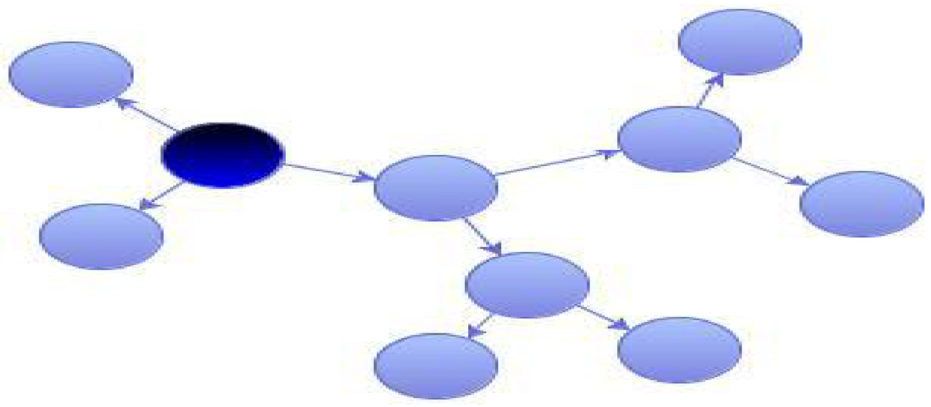


Figure 8.3: Randomly selected node will disseminate INIT packet in the network from reference [1]

- ECHO Packet: This is a value that is disseminated over the network using the Drip protocol and is same as the node id of the node being the source of the dissemination, hence if there is any value other than 0 received by any node it is regarded as an ECHO packet.

## 8.1 Results

The algorithm was simulated for a varied number of nodes which were 10, 25, 50 and 100 in number. The chief parameter of the node that was analyzed was its Power and the time consumed for the entire algorithm to execute.
Energy was measured using PowerTOSSIM-Z and was then graphically plotted as below:

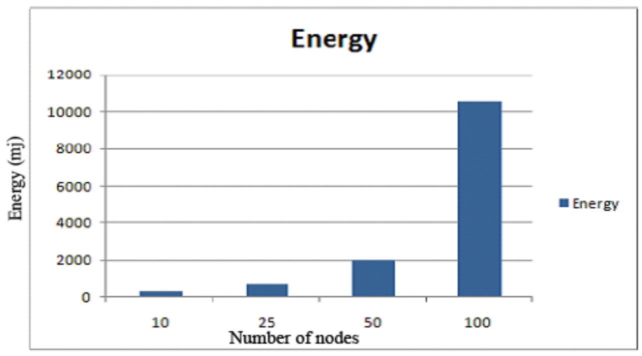


Figure 8.4: Energy vs. Number of Nodes

The energy has been measured in millijoules (mj). Y-axis in the figure 8.4 indicates energy consumed by the nodes while X-axis indicates number of nodes used for simulation. We can discern from the figure 8.4 that as the number of nodes increases, the total energy consumption of the nodes also increases. As more and more nodes are added, the energy consumed increases non-linearly.

The Time consumed during the algorithm was measured and plotted as below:

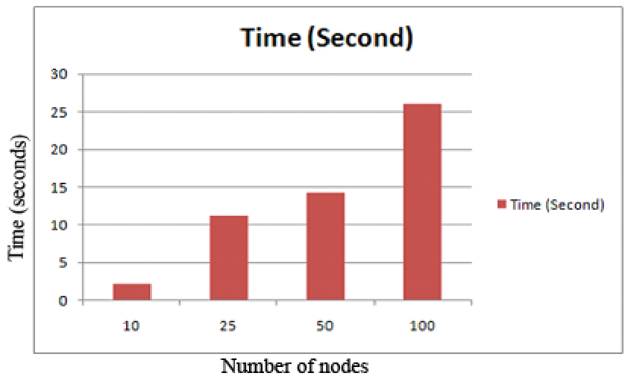


Figure 8.5: Time vs. Number of Nodes

Y-axis in the figure 8.5 indicates the time taken by the nodes for simulation while X-axis indicates the number of nodes used for simulation. As the number of nodes increases, figure 8.5 clearly shows the increase in the amount of time consumed by the algorithm in seconds.

Figure 8.6 shows charge versus time.Y axis in the figure 8.6 indicates charge which is measured in mAh.And X axis shows the time in seconds.As time increases the rate of a charge of a node also decreases.Here rate of charge decreases more in CTP as compared to broadcast fault tolerant algorithm in multi-hop network using RSSI.

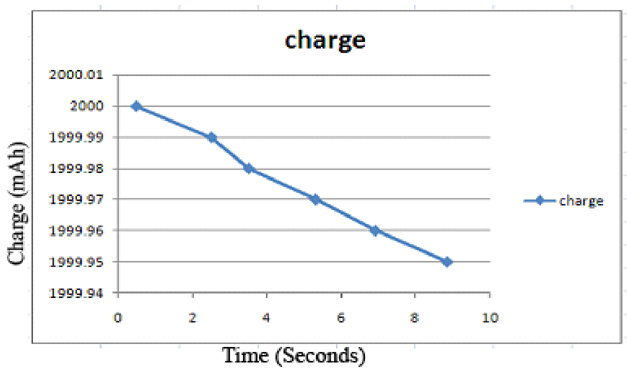


Figure 8.6: Charge vs.Time

# Chapter 9

# Broadcast Fault Tolerant Algorithm in Multi-Hop Network using RSSI

In multi-hop network ,global broadcasting is not possible because all nodes are not directly connected to each other. In multi-hop networks local broadcast is possible in which each node sends packet to its neighbors.In this approach broadcast fault tolerant algorithm is integrated with RSSI.RSSI means power received by antenna. It is also used to determine amount of radio energy present in the radio channel. Once the network is cleared for sending the data to sender node, it will send its packet to destination and the packet is sent to destination hop by hop. Once the packet is reached to destination, it will measure its RSSI value and also the signal strength of wireless network at different power levels.

Here the whole protocol is divided in different rounds:

- In first round, each node locally broadcasts HELO packet.
- In second round, receiving HELO packet from neighbor node sends ACK packet to sender node. After receiving ACK packet node measures its RSSI value and stores all necessary information. Each node also stores which node can be selected as its parent.
- In third round, information collected from the previous stage is sent to base station and base station processes this information and creates routing table.
- In fourth round, base station sends pruned routing table to all nodes.

Pruning refers to deleting unnecessary entries from routing table. When routing table is constructed at base station, it sends this routing table to all nodes in the network. When it forwards this table to nodes, they will prune entries of nodes which are at same level or higher level in the given graph. This will enable nodes to communicate efficiently. Using this table each node will know how much power is required to transmit a packet and for global broadcast to which node it should send packet. This method will allow power efficient global broadcast in the whole wireless sensor network.The concept of RSSI (power based routing) makes this approach different from previous approaches. As power is a very important resource in sensor networks and this approach uses the least power.

The algorithm is divided in different stages:

- First Stage: All nodes send HELO packet to base station using CTP and base station finds total number of nodes in network as shown below.

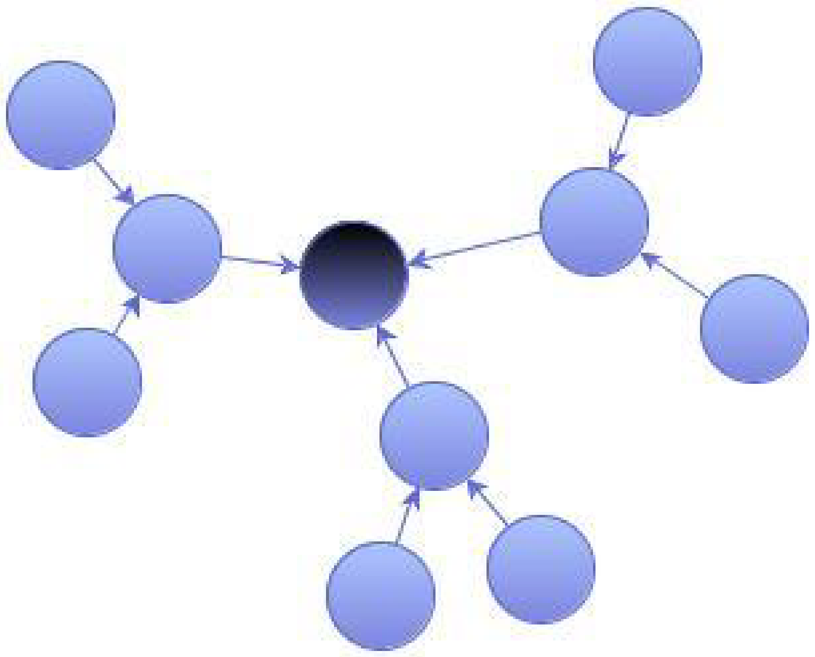


Figure 9.1: Nodes sending HELO packet to base station from reference [1]

- Second Stage: Base station sends "n (number of nodes)" to all nodes using Dissemination as shown below.

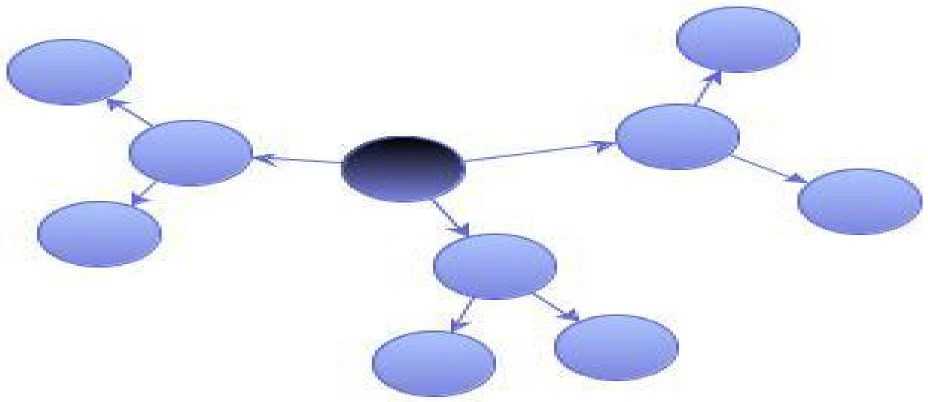


Figure 9.2: Base station sending number of nodes in the network to all nodes from reference [1]

- Third Stage: Each node sends HELO packet to its neighbor as shown below.

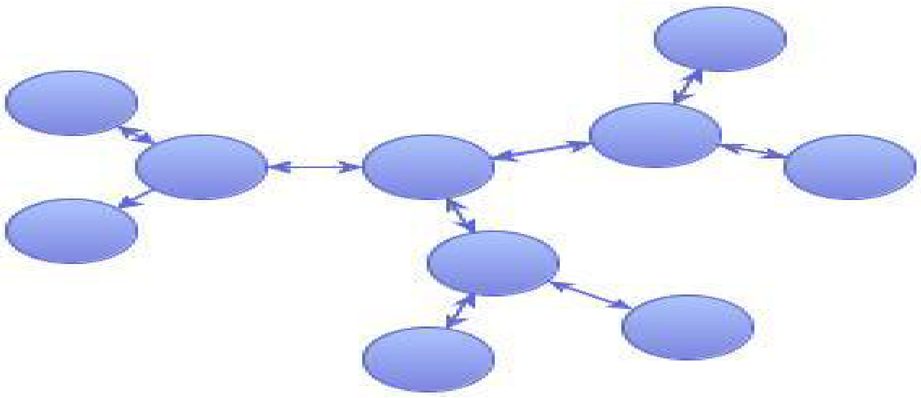


Figure 9.3: Nodes sending HELO packet to each other from reference [1]

- Fourth Stage: After receiving HELO packet from neighbor node sends ACK packet to sender node. After receiving ACK packet node measures its RSSI value and stores all necessary information. Each node also stores which node can be selected as its parent.

- Fifth Stage: Information collected from the previous stage is sent to base station using FRNDS packet and base station processes this information and creates routing table.

- Sixth Stage: Base station sends pruned routing table to all nodes.
- Seventh Stage: In this stage one node will be randomly selected and will disseminate INIT packet using pruned routing table.
- Eighth Stage: After receiving INIT packet, nodes send ECHO packet. Nodes keep counter of ECHO packets.
- Ninth Stage: If any node has received n-f ECHO packets in previous round then it establishes successful communication.

Packet Structures:

- HELO Packet: The HELO packet is a 16bit packet that contains the node id of the node that is transmitting the packet. Through this the base station or the root of the tree in the Collection Tree Protocol receives the node id of the node that sent it the HELO packet.
- INIT Packet: The INIT packet is a 32 bit packet that contains the node id of the node that is transmitting INIT.
- ECHO Packet: It is a 32 bit packet that is disseminated over the network as a reply for the INIT packet.
- FRNDS Packet: It is a packet which carries information about nodes neighbors and power required to send packet to its neighbors.

## 9.1 Results

The algorithm was simulated for a varied number of nodes which were 10, 25, 50 and 100 in number. The chief parameter of the node that is analyzed is its Power and the time consumed for the entire algorithm to execute.
Energy is measured using PowerTOSSIM-Z and is then graphically plotted as below:

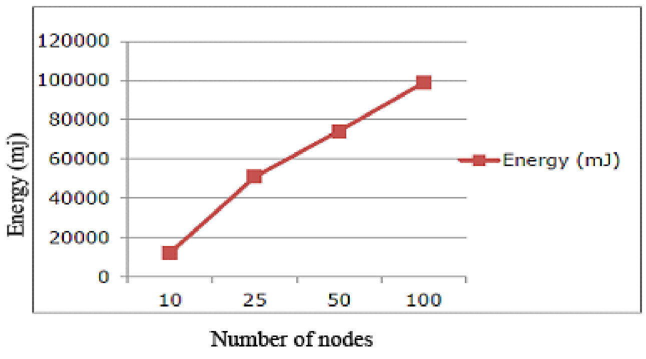


Figure 9.4: Energy vs. Number of Nodes

The energy has been measured in millijoules (mj). Y-axis in the figure 9.4 indicates Energy consumed by the nodes while X-axis indicates number of nodes used for simulation. We can discern from the figure 9.4 that as the number of nodes increases, the total energy consumption of the nodes also increases. As more and more nodes are added, the energy consumed increases non-linearly.

The Time consumed during the algorithm was measured and plotted as below:

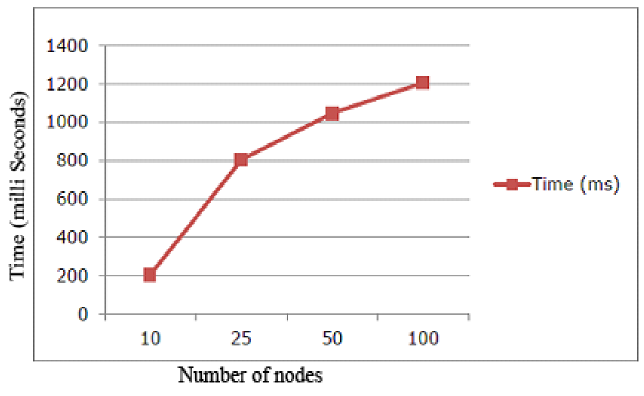


Figure 9.5: Time vs. Number of Nodes

Y-axis in the figure 9.5 indicates the time taken by the nodes for simulation while X-axis indicates the number of nodes used for simulation. As the number of nodes increases, figure 9.5 clearly shows the increase in the amount of time consumed by the algorithm in seconds.

Figure 9.6 shows charge versus time.Y axis in the figure 9.6 indicates charge which is measured in mAh.And X axis shows the time in seconds.As time increases the rate of a charge of a node also decreases.Here rate of charge decreases less in broadcast fault tolerant algorithm in multi-hop network using RSSI as compared to CTP.

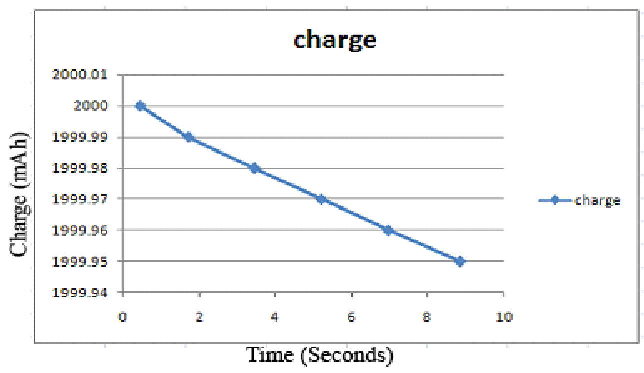


Figure 9.6: Charge vs.Time

# Chapter 10

# Implementation of CTP over Castalia

CTP suffers a problem by disseminating data statically in network and so it is not able to send data to the node that it does not belong to its range. Mobility poses challenges in CTP to dynamically route data. So by integrating line mobility model in CTP it dynamically send data to sink in Castalia simulator and evaluate the performance of CTP as applied in mobile wireless sensor network scenarios. It is a simulator for wireless sensor network. It is based on the OMNET++ platform. It provides various features like advanced channel model based on empirically measured data.

- It defines a map of path loss and also connections between nodes.
- It contains advanced radio model based on real radios for low-power communication.
- Multiple TX power levels with individual node variations allowed.
- Extended sensing modeling provisions.
- Highly flexible physical process model.
- Sensing device noise and power consumption.
- Node clock drift.
- MAC and routing protocols available.

It has modular structure with many interconnecting modules. It has modules which has one or more parameters that will affect its behavior. An NED file defines the basic structure of a module by defining its input/output gates and its parameters. In the NED file we can also define default values for the parameters. OMNET will allow to select configurations defined in a configuration file named as omnet.ini file.

OMNET's basic concepts are modules and messages. A simple module is the basic unit of execution. It accepts messages from other modules or itself, and according to the message, it executes a piece of code. The code can keep state that is altered when messages are received and can send or schedule new messages. There are also composite modules. A composite module is just a construction of simple and/or other composite modules. Proposed Approach : Line mobility model integrated with CTP: The Line mobility module moves the node in a straight line segment.The straight line segment defines the starting location of node and destination location given by the xCoorDestination, yCoorDestination parameters of the mobility module, also mobility speed is included in omnet.ini file. The distance between nodes is calculated as below:

```
distance = sqrt(pow(loc1_x - loc2_x, 2) + pow(loc1_y - loc2_y, 2)
```

It also explains that how much distance a node should move which is calculated with the help of update interval as shown below. It will have some update interval which will be updated accordingly when a node moves its position.

```
double tmp = (distance / speed) / updateInterval);
incr_x = (loc2_x - loc1_x) / tmp;
incr_y = (loc2_y - loc1_y) / tmp;
incr_z = (loc2_z - loc1_z) / tmp;
setLocation(loc1_x, loc1_y, loc1_z);
scheduleAt(simTime() + updateInterval);
```

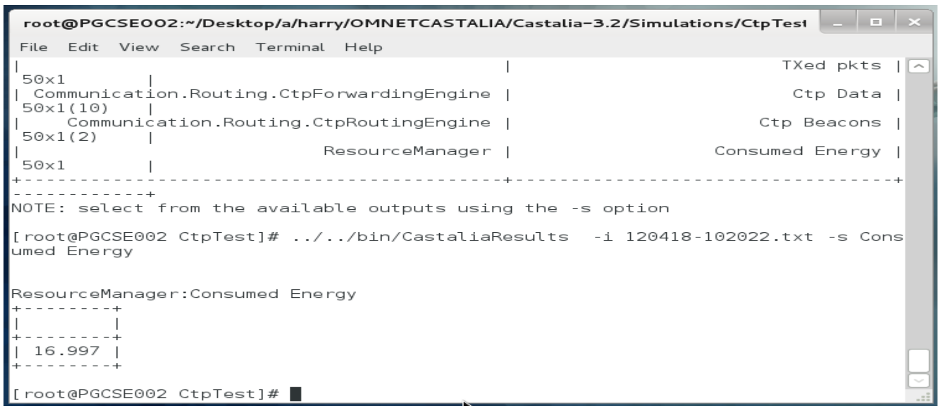
Line mobilty module was integrated in CTP omnet.ini configuration file. The protocol was simulated for a varied number of nodes which were 50,100,150,200,250 in number. And comparison between Existing and modified CTP is shown.

Table 10.1: Simulation Result of scenario upto 250 nodes

| Number of nodes | Simulation time(Seconds) | Field size | Radius(Meters) | Existing CTP(Joules) | Modified CTP(Joules) |
|---|---|---|---|---|---|
| 50 | 200 | 100*100 | 25 | 16.997 | 13.257 |
| 100 | 200 | 150*150 | 25 | 20.395 | 16.732 |
| 150 | 200 | 200*200 | 25 | 23.793 | 20.395 |
| 200 | 200 | 250*250 | 25 | 27.191 | 23.577 |
| 250 | 200 | 300*300 | 25 | 30.58 | 26.98 |

Simulation Results

Existing CTP:

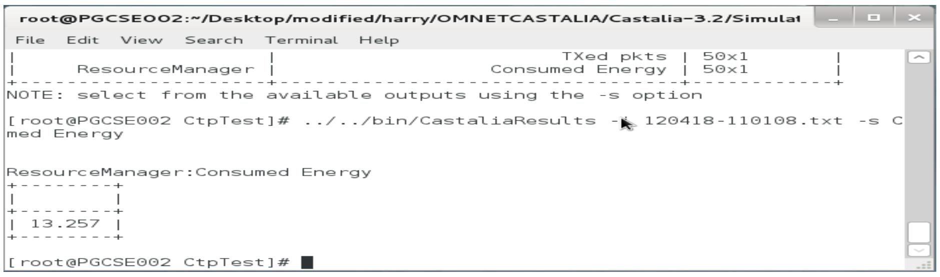


```
root@PGCSE002:~/Desktop/a/harry/OMNETCASTALIA/Castalia-3.2/Simulations/CtpTest
File  Edit  View  Search  Terminal  Help
|                                          |                                TXed pkts |
 50x1         |
| Communication.Routing.CtpForwardingEngine |                                Ctp Data |
 50x1(10)     |
|     Communication.Routing.CtpRoutingEngine |                             Ctp Beacons |
 50x1(2)      |
|                             ResourceManager |                         Consumed Energy |
 50x1         |
+------------------------------------------+---------------------------------+
------------+
NOTE: select from the available outputs using the -s option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-102022.txt -s Cons
umed Energy


ResourceManager:Consumed Energy
+---------+
|         |
+---------+
| 16.997 |
+---------+

[root@PGCSE002 CtpTest]# █
```

Figure 10.1: For nodes 50

Figure 10.2:

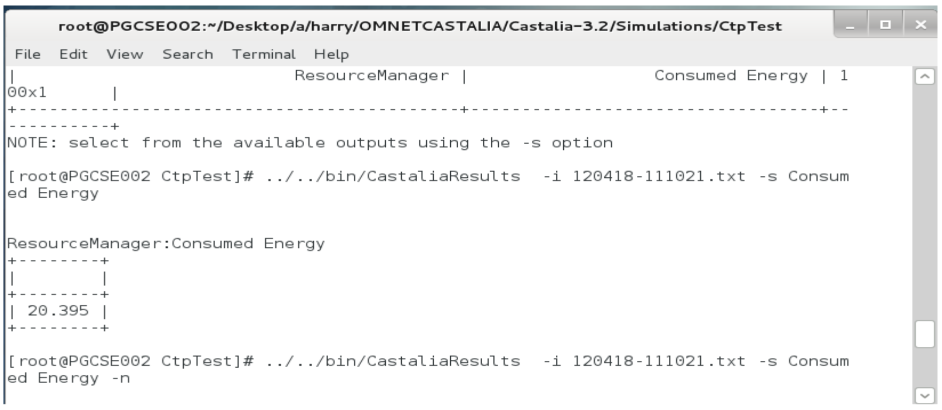
Modified CTP:



```
root@PGCSE002:~/Desktop/modified/harry/OMNETCASTALIA/Castalia-3.2/Simulat
File  Edit  View  Search  Terminal  Help
|                    |                              TXed pkts | 50x1        |
|     ResourceManager |                        Consumed Energy | 50x1        |
+--------------------+------------------------------------+-----------+
NOTE: select from the available outputs using the -s option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults -i 120418-110108.txt -s C
med Energy


ResourceManager:Consumed Energy
+---------+
|         |
+---------+
| 13.257 |
+---------+

[root@PGCSE002 CtpTest]# █
```

Figure 10.3: For nodes 50

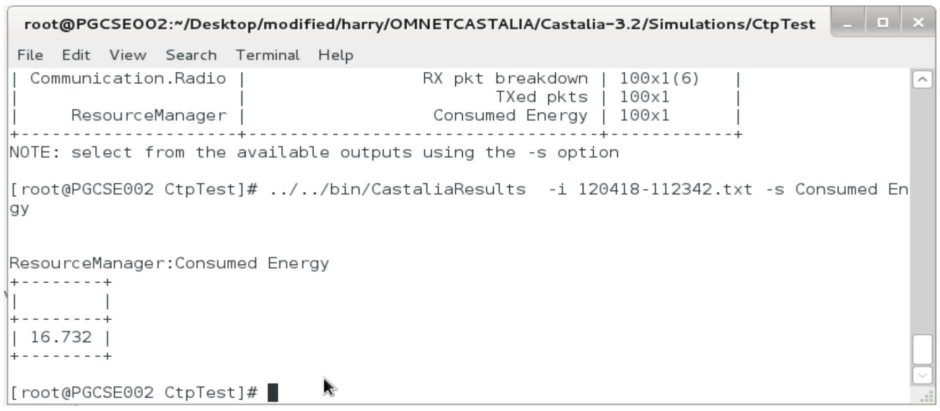
Existing CTP:


root@PGCSE002:~/Desktop/a/harry/OMNETCASTALIA/Castalia-3.2/Simulations/CtpTest
File Edit View Search Terminal Help

```
|                           ResourceManager |                 Consumed Energy | 1
00x1       |
+-------------------------------------------+---------------------------------+--
----------+
NOTE: select from the available outputs using the -s option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-111021.txt -s Consum
ed Energy


ResourceManager:Consumed Energy
+--------+
|        |
+--------+
| 20.395 |
+--------+

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-111021.txt -s Consum
ed Energy -n
```

Figure 10.4: For nodes 100

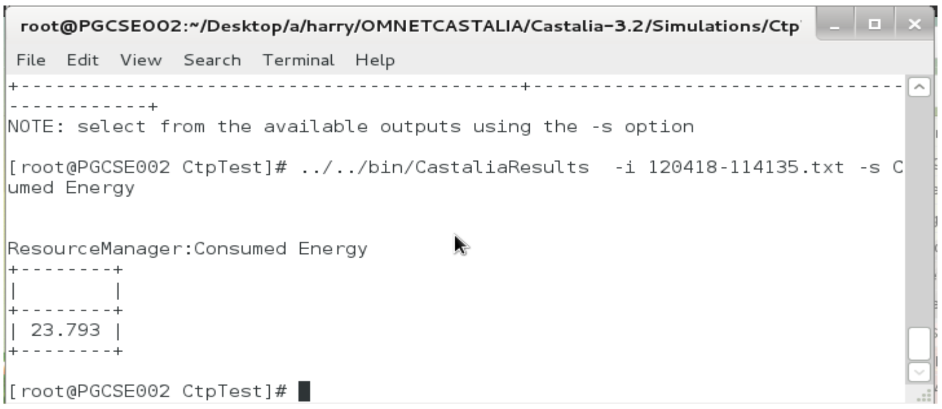
Modified CTP:


root@PGCSE002:~/Desktop/modified/harry/OMNETCASTALIA/Castalia-3.2/Simulations/CtpTest
File Edit View Search Terminal Help

```
| Communication.Radio |                  RX pkt breakdown | 100x1(6)   |
|                     |                         TXed pkts | 100x1      |
|     ResourceManager |                   Consumed Energy | 100x1      |
+---------------------+-----------------------------------+------------+
NOTE: select from the available outputs using the -s option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-112342.txt -s Consumed En
gy


ResourceManager:Consumed Energy
+--------+
|        |
+--------+
| 16.732 |
+--------+

[root@PGCSE002 CtpTest]# 
```

Figure 10.5: For nodes 100

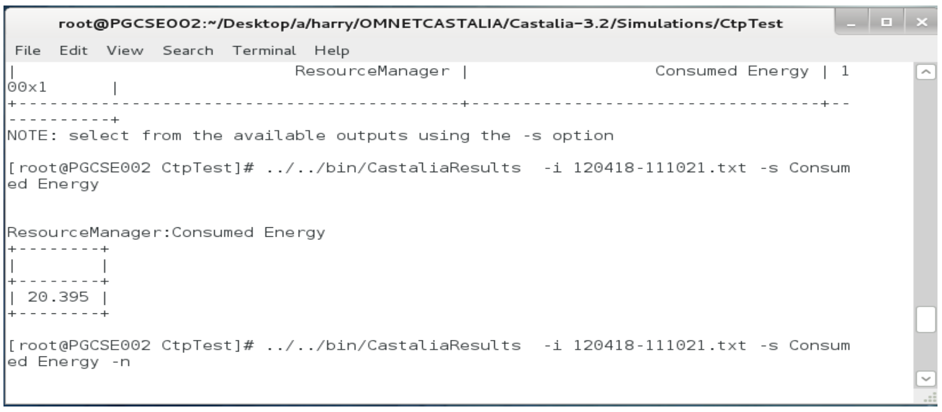
Existing CTP:


root@PGCSE002:~/Desktop/a/harry/OMNETCASTALIA/Castalia-3.2/Simulations/Ctp
File Edit View Search Terminal Help

```
+----------------------------------------+--------------------------------
------------+
NOTE: select from the available outputs using the -s option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-114135.txt -s C
umed Energy


ResourceManager:Consumed Energy
+--------+
|        |
+--------+
| 23.793 |
+--------+

[root@PGCSE002 CtpTest]# 
```

Figure 10.6: For nodes 150

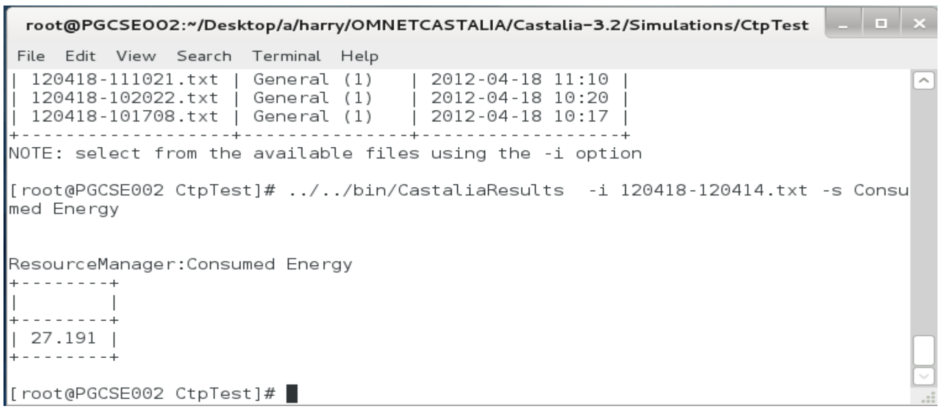
Modified CTP:


root@PGCSE002:~/Desktop/a/harry/OMNETCASTALIA/Castalia-3.2/Simulations/CtpTest
File Edit View Search Terminal Help

```
|                           ResourceManager |                 Consumed Energy | 1
00x1      |
+----------------------------------------+----------------------------------+--
----------+
NOTE: select from the available outputs using the -s option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-111021.txt -s Consum
ed Energy


ResourceManager:Consumed Energy
+--------+
|        |
+--------+
| 20.395 |
+--------+

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-111021.txt -s Consum
ed Energy -n
```

Figure 10.7: For nodes 150

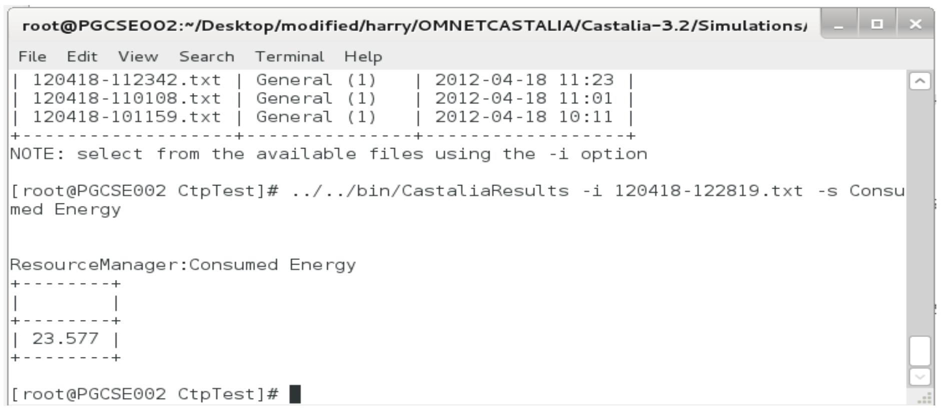
Existing CTP:


root@PGCSE002:~/Desktop/a/harry/OMNETCASTALIA/Castalia-3.2/Simulations/CtpTest
File Edit View Search Terminal Help

```
| 120418-111021.txt | General (1)   | 2012-04-18 11:10 |
| 120418-102022.txt | General (1)   | 2012-04-18 10:20 |
| 120418-101708.txt | General (1)   | 2012-04-18 10:17 |
+-------------------+---------------+------------------+
NOTE: select from the available files using the -i option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120418-120414.txt -s Consu
med Energy


ResourceManager:Consumed Energy
+--------+
|        |
+--------+
| 27.191 |
+--------+

[root@PGCSE002 CtpTest]#
```

Figure 10.8: For nodes 200

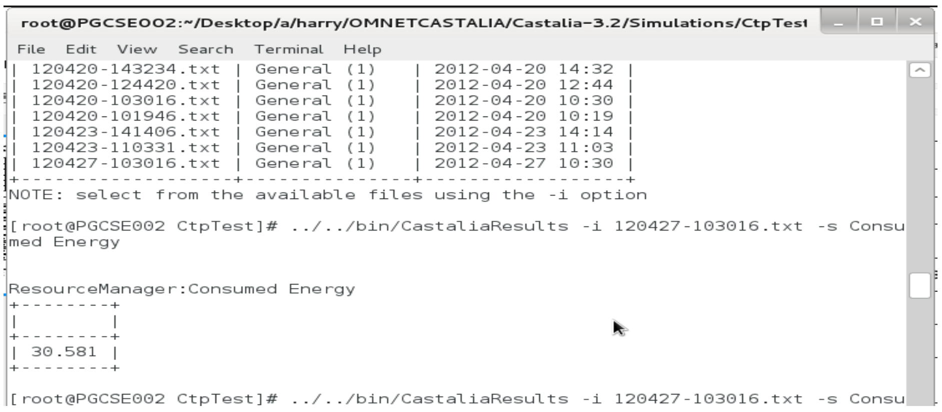
Modified CTP:


root@PGCSE002:~/Desktop/modified/harry/OMNETCASTALIA/Castalia-3.2/Simulations/
File Edit View Search Terminal Help

```
| 120418-112342.txt | General (1)   | 2012-04-18 11:23 |
| 120418-110108.txt | General (1)   | 2012-04-18 11:01 |
| 120418-101159.txt | General (1)   | 2012-04-18 10:11 |
+-------------------+---------------+------------------+
NOTE: select from the available files using the -i option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults -i 120418-122819.txt -s Consu
med Energy


ResourceManager:Consumed Energy
+--------+
|        |
+--------+
| 23.577 |
+--------+

[root@PGCSE002 CtpTest]#
```

Figure 10.9: For nodes 200

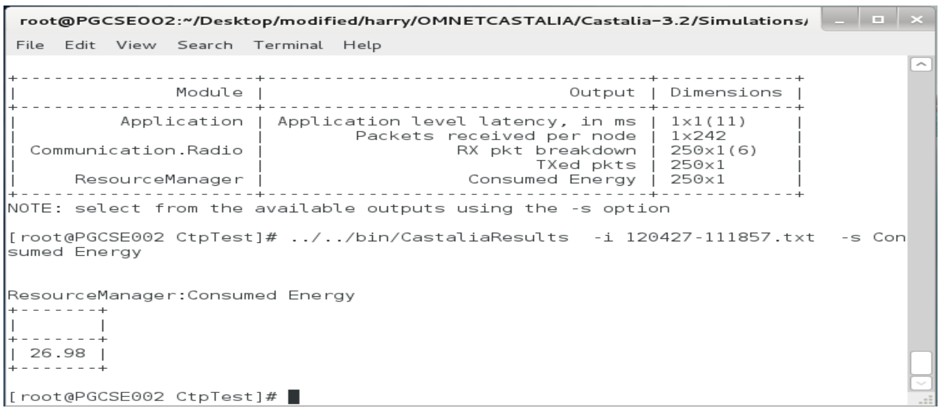
Existing CTP:


root@PGCSE002:~/Desktop/a/harry/OMNETCASTALIA/Castalia-3.2/Simulations/CtpTest
File Edit View Search Terminal Help

```
| 120420-143234.txt | General (1)    | 2012-04-20 14:32 |
| 120420-124420.txt | General (1)    | 2012-04-20 12:44 |
| 120420-103016.txt | General (1)    | 2012-04-20 10:30 |
| 120420-101946.txt | General (1)    | 2012-04-20 10:19 |
| 120423-141406.txt | General (1)    | 2012-04-23 14:14 |
| 120423-110331.txt | General (1)    | 2012-04-23 11:03 |
| 120427-103016.txt | General (1)    | 2012-04-27 10:30 |
+-------------------+----------------+------------------+
NOTE: select from the available files using the -i option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults -i 120427-103016.txt -s Consu
med Energy


ResourceManager:Consumed Energy
+--------+
|        |
+--------+
| 30.581 |
+--------+

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults -i 120427-103016.txt -s Consu
```

Figure 10.10:   For 250 nodes

Modified CTP:


root@PGCSE002:~/Desktop/modified/harry/OMNETCASTALIA/Castalia-3.2/Simulations/
File Edit View Search Terminal Help

```
+---------------------+-----------------------------------+------------+
|              Module |                            Output | Dimensions |
+---------------------+-----------------------------------+------------+
|         Application | Application level latency, in ms  | 1x1(11)    |
|                     |          Packets received per node | 1x242      |
| Communication.Radio |                 RX pkt breakdown  | 250x1(6)   |
|                     |                        TXed pkts  | 250x1      |
|     ResourceManager |                   Consumed Energy | 250x1      |
+---------------------+-----------------------------------+------------+
NOTE: select from the available outputs using the -s option

[root@PGCSE002 CtpTest]# ../../bin/CastaliaResults  -i 120427-111857.txt  -s Con
sumed Energy


ResourceManager:Consumed Energy
+-------+
|       |
+-------+
| 26.98 |
+-------+

[root@PGCSE002 CtpTest]#
```

Figure 10.11:   For 250 nodes

Resource manager in Castalia defines the energy calculation and also defines the initial energy of a node. This module is declared in routing engine.cc file in CTP which calculates energy whenever a node sends data to the neighbour having minimum ETX value.

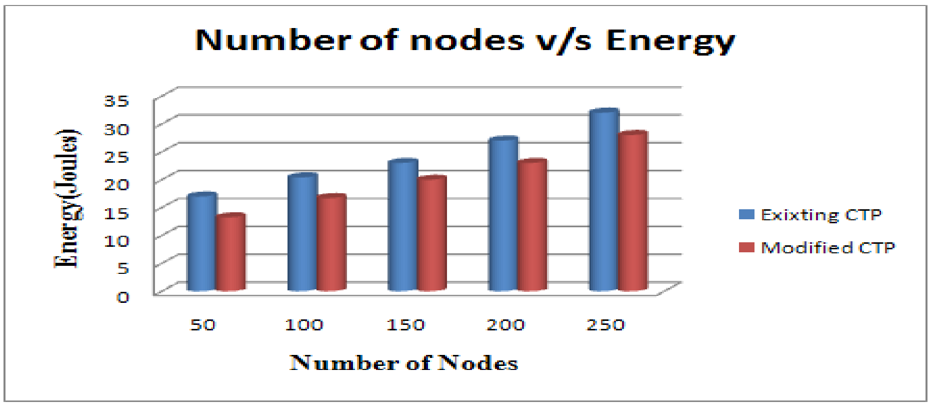


Figure 10.12: Number of nodes v/s Energy

Figure 10.12 shows the energy which is measured in joules and is simulated from 50 to 250 nodes. It indicates that modified CTP consumed less energy than existing CTP. In an existing CTP, packets are lost more due to network congestion then modified CTP.

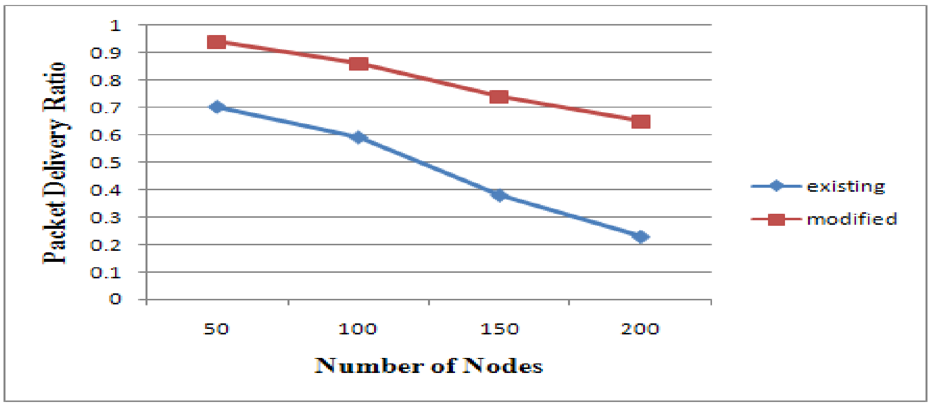


Figure 10.13: Packet delivery ratio

Figure 10.13 shows the packet delivery ratio which is simulated from 50 to 250 nodes and indicates that modified CTP has higher packet delivery ratio than existing CTP by 32.3 percentage, because of communication radius is small the nodes are more closer to each other and so packets are delivered more to sink node.

# Chapter 11

# Conclusion

Energy is an important constraint in wireless sensor network.So to optimize energy in collection tree protocol, I have implemented broadcast fault tolerant algorithm on multi-hop networks using CTP and multi-hop networks using RSSI, then compare the results of these two different topologies. Collection tree protocol disseminates data statically so to dynamically disseminate data line mobility module is integrated in existing CTP protocol over castalia wireless sensor network simulator. And compare the simulation results of an existing CTP and modified CTP which indicates that by including line mobility module it sends data to sink node dynamically and because of small communication radius between nodes, packet delivery ratio is higher in modified CTP by 32.3 percentage. Also modified CTP consumed less energy than existing CTP.

# References

[1] Shashank Juyal, Herat Gandhi, Shatadru Chattopadhyay,Gaurang Raval, "Implementation of Byzantine Fault Tolerance Algorithm in Multi-Hop Networks",Institute of Technology, Nirma University Ahmedabad, Gujarat, India,2011.

[2] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss and Philip Levis, "Collection Tree Protocol", Proceedings of the 7th ACM Conference: Embedded Networked Sensor Systems , New York, NY,USA, 2009.

[3] Yongjun Li, Hu Chen, Rongchuan He, Rong Xie*, Shaocong Zou, "ICTP: An Improved Data Collection Protocol Based On CTP",*wireless communication and signal processing,*pp. 1-5,Oct-2010.

[4] Jian Wan, Xin Luo, Xianghua Xu,"Research on Synchronous Low Power Listening for Collection Tree Protocol in WSN",IEEE Asia-Pacific Services Computing Conference,2010,pp.6-10.

[5] Sundeep Pattem, Godwin Shen, Ying Chen, Bhaskar Krishnamachari, Antonio Ortega "SenZip: An Architecture for Distributed En-route Compression in Wireless Sensor Networks",in ESSA University of Southern California, Los Angeles, CA, USA,april-2009.

[6] CTP protocol
http://www.tinyos.net/tinyos-2.x/tos/lib/net/ctp/

[7] Jinfu Zheng,"Simulate the routing path selection of mobile motes in CTP protocol", Project report for ENGR 5720G.

[8] Ugo Colesanti, Silvia Santini,"A Performance Evaluation Of The Collection Tree Protocol Based On Its Implementation For The Castalia Wireless Sensor Networks

Simulator", Technical Report Nr. 681Department of Computer Science ETH Zurich August 31, 2010.

[9] Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, Omprakash Gnawali, "Routing Without Routes: The Backpressure Collection Protocol",University of Southern California, Los Angeles, CA Stanford University, Stanford.

[10] V. D. Park, M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks",IEEE INFOCOM, VOL3,1997,pp.1405-1413.

[11] Shuai Gao and Hongke Zhang,Sajal Das,"Efficient Data Collection in Wireless Sensor Networks With Path-constrained Mobile Sinks",IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks,2009,pp.1-9.

[12] Tzung-Shi Chen,Hua-Wen Tsai,Chih-Ping Chu, "Gathering-Load-Balanced Tree Protocol for Wireless Sensor Networks", Sensor Networks, IEEE International Conference on Ubiquitous, and Trustworthy Computing, 2006,pp.8-13.

[13] TinyOS wiki TEP119-collection
http://www.tinyos.net/tinyos-2.x/doc/txt/tep119.txt

[14] Wen Hu, Brano Kusy, Christian Richter, Michael Bruenig, Cong Huynh,"Demo Abstract: Radio-diversity Collection Tree Protocol",10th International Conference on Information Processing in Sensor Networks (IPSN),2011,pp.111-112.

[15] Yuri Tselishchev, Athanassios Boulis, Lavy Libman, "Experiences and Lessons from Implementing a Wireless Sensor Network MAC Protocol in the Castalia Simulator", submitted to IEEE Wireless Communications and Networking Conference 2010 (WCNC 2010), Sydney, Australia.

# Index