Distributed Hash Table For Scalable Wireless Sensor Network

By

Valay D Shah

10MCEC27

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2012

Distributed Hash Table For Scalable Wireless Sensor Network

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Valay D Shah

10MCEC27

Guide

[Prof. Gaurang Raval]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2012

Undertaking for Originality of the Work

I, **Valay D Shah**, **10MCEC27**, give undertaking that the Major Project entitled **"Distributed Hash Table For Scalable Wireless Sensor Network"** 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.

Signature Of Student:_____

Date:_____

Place:Nirma University Ahmedabad

Endorsed by

(Signature of Guide)

Dedication

This thesis is dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time. who taught me that the best kind of knowledge to have is that which is learned for its own sake.

I would also like to dedicate this thesis to my family, who passed on a love of reading and respect for education, who have never failed to give me moral support, for giving all my need during the time I developed system.

Certificate

This is to certify that the Major Project entitled "Distributed Hash Table For Scalable Wireless Sensor Network" submitted by Valay D Shah (10MCEC27), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University of Science and Technology, 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.

Dr. S.N. Pradhan
Professor and Co-ordinator
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. Gaurang Raval
Guide and Asso Professor,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K. Kotecha
Director,
Institute Of Technology,
Nirma University, Ahmedabad

Abstract

In recent years, wireless sensor networks (WSNs) has gained a lot of attention in both research and application fields. A fundamental goal of researches in WSNs is to reduce the communication operations and prolong the total lifetime of sensor networks. Therefore, researchers are focusing on optimal routing techniques which are one of the main factors to reduce energy usage of sensor nodes and wireless sensor networks. Distributed Hash Tables (DHTs) over WSNs is a new routing paradigm promises several advantages over conventional routing protocols. This thesis presents an overview of using DHTs in WSNs, especially routing techniques using DHTs over WSNs such as GHT, CSN, CHR, T-DHT, VRR and ScatterPastry and how can we transmit packets over multiple paths with Round Robin Algorithm to improve the Network Life Time.We have implemented Round Robin Algorithm over multiple paths and compared various implementations with parameters like Network Life Time, Throughput, End to End Delay.

Index Terms–Distributed Hash Tables, Wireless Sensor Networks, routing, DHTs over WSNs.

Acknowledgements

With immense pleasure, I would like to present this report on the dissertation work related to "Distributed Hash Table For Scalable Wireless Sensor Network". I am very thankful to all those who helped me for the successful completion of the dissertation and for providing valuable guidance throughout the project work.

I would first of all like to offer thanks to **Dr. S. N. Pradhan** & Programme Co-ordinator M.Tech. CSE, Institute of Technology, Nirma University, Ahmedabad and **Prof.Gaurang Raval** Guide M.tech CSE, Institure of Technology, Nirma University, Ahmedabad whose keen interest and excellent knowledge base helped me to finalize the topic of the dissertation work. Their constant support and interest in the subject equipped me with a great understanding of different aspects of the required architecture for the project work. They have shown keen interest in this dissertation work right from beginning and have been a great motivating factor in outlining the flow of my work.

My sincere thanks and gratitude to **Prof. D.J. Patel**, Professor and Head, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his continual kind words of encouragement and motivation throughout the Dissertation work.

I am thankful to Nirma University for providing all kind of required resources. I would like to thank The Almighty, my family, especially my wife, for supporting and encouraging me in all possible ways. I would also like to thank all my friends who have directly or indirectly helped in making this dissertation work successful.

- Valay D Shah

10MCEC27

Abbreviation Notation and Nomenclature

GPSR ..... Greedy Perimeter Routing
GHT ..... Geographic Hash Table
CHR ..... Cell Hash Routing
DHT ..... Distributed Hash Table
GAF ..... Geographic Adaptive Fidelity
RR ..... Round Robin

Contents

List of Figures

Chapter 1

Introduction

P2P Network is the network without the notion of clients or servers but only equal peer nodes that simultaneously function as both client and servers. Wireless sensor network is peer to peer network, most mobile peer to peer approaches deploy an unstructured peer to peer network on top of ad-hoc routing protocol, such as AODV or DSR ,thus they suffer from the limited scalability of ad-hoc routing protocols and the limited scalability of unstructured network as both layers strongly rely on flooding. So some scalable routing protocol is needed for wireless sensor network. In recent years, wireless sensor networks (WSNs) has gained a lot of attention in both research and application fields. A fundamental goal of researches in WSNs is to reduce the communication operations and prolong the total lifetime of sensor networks. Therefore, researchers are focusing on optimal routing techniques which are one of the main factors to reduce consumable energy of sensor and wireless sensor networks. Distributed Hash Tables (DHTs) over WSNs is a new routing paradigm promises several advantages over conventional routing protocols. This paper presents an overview of using DHTs in WSNs, especially routing techniques using DHTs over WSNs such as GHT, CSN, CHR, T-DHT, VRR and Scatter Pastry. After that we compare with methods which used DHTs in WSNs with three main parameters such as Scalability, Energy-efficient, and Data storage/lookup Efficiency.

DHT (Distributed Hash Table): It maps Data items, key-value pairs, on node IDs .key is computed via hash function from string describing data item. Node ID is derived from its Geographical position. Properties of DHT:

- High scalability
- High robustness against frequent peers departures and arrivals
- Self organization: No need for a central server
  1.No single-point failure problem
  2. Higher fault tolerance

Why DHT in peer to peer Network?

- Balanced distribution of data among nodes: Avoid having nodes overloaded with data
- Minimum disruption by nodes joining, leaving and failing: Only a small part of the network concerned by a change in the set of Participants
  1. Consistency

1.1 Background

In the last few years, an increasing number of massively Peer to Peer (P2P) systems with millions of participants have emerged within very short time frames. P2P systems have been developed and applied in a lot of fields such as file sharing, search, grid computing, content distribution and video streaming. During its short history, P2P passed through three basic strategies to store and retrieve data in distributed systems: centralized servers, flooding search, and distributed indexing- Distributed Hash Tables (DHTs). Distributed hash tables (DHTs) are third generation P2P protocols forming structured overlays which have several advantages compared to other

P2P paradigms. In a DHT, nodes must cooperate to maintain the coherence of the data. Each node stores a partial view of the whole distributed system which effectively distributes the routing information. Nodes use a consistent hash function to determine the peer that holds a given value. DHTs are algorithms used in modern peer-to-peer applications, which provide a reliable, scalable, fault tolerant and efficient way to manage P2P networks in a true peer to peer manner. Since DHTs are successful decentralized, scalable and self-organized P2P systems and, they motivate researchers to build a routing overlay over WSNs.

1.2 Objective of Study

Apart from the above mentioned goals and vision, this report addresses the specific aspect of Scalable Distributed Hash Table In Wireless Sensor Network . The experimental setup has been created on a computer, with software based sensor node. Hence, the objective of the dissertation work can be summarized as:

Study the working of different types of routing using DHT over wireless sensor Network and Study the scalability of various types of Routing using DHT over Wireless Sensor Network.

Next objective is to find out which are the problems in design of all these routing algorithms and what are the open research problems for them newline
Next objective is to improve routing using DHT algorithm in such way that it become more scalable and increase network life time and implementation of Routing algorithms on NS2 and compare the result with other existing Routing protocols.

1.3 Scope of Work and Motivation

The experimental setup, prepared for the dissertation work, includes a simulator installed on a single computer in a Computer Lab. A part of the work is study P2P overlay in WSNs or DHTs based routing in WSNs can be useful in several ways such as efficient data lookup, guaranties on lookup times, location independence, overlay application, and so on. As we know that the main purpose of WSNs is to collect and storage data about phenomena of interest. Thus, looking up or reporting data of interest in one of the main operations in WSNs, and optimizing this operation is a fundamental research problem. DHTs provide near optimum data lookup times that could help solve the efficient data lookup problem in WSNs. Additionally, DHTs based data lookup operation is the only core function of P2P overlay. Thus, it has been used to build a wide variety of applications such as file systems, event notification, content distribution, and so on.

In recent years, P2P systems have been growing in popularity rapidly. They are applied to a lot of fields. Especially, P2P and WSNs systems share some common features such as self-organizing, scalable and decentralized. Beside, they too share a common problem that is Churn (the rate of nodes arrival and departure). With successfulness of DHTs in P2P systems, they are motivating researchers to built routing algorithms over WSNs.

Chapter 2

Data Centric Storage

This chapter describe need for Data Centric Storage. We now describe[2] how the information might be extracted from the sensor networks. **Task, Action and Queries** Users send instructions (by flooding or some other global dissemination method) to sensor nodes to run certain local identification tasks. These tasks could be simple, such as taking temperature readings, or complex, such as identifying an animal from a collection of sensor readings. In essence, one can think of tasks as downloaded code. Once an event has been identified, nodes can take a number of different actions. For example, a node could be instructed to send event information to external storage, or to store the event information locally, or to use data-centric storage. How queries are executed will depends on the actions nodes take upon event detection. If event information is stored locally then queries must be flooded to all nodes (unless the user has prior knowledge about the location of the event). If event information is stored using data-centric storage, the query can be sent to the sensor network node associated with that event name. Note that queries can be either externally generated by users or internally generated by sensor network nodes engaged in event detection

2.1 Storage Type

- External Storage (ES): Upon detection of events, the relevant data is sent to external storage where it can be further processed as needed. This entails a cost of O($\sqrt{n}$) for each event. There is no cost for external queries since the event information is already external; queries generated by internal nodes in the process of doing event detection will incur a cost of O($\sqrt{n}$) to reach the external storage.

- Local Storage (LS): Event information is stored locally (at the detecting node) upon detection of an event; this incurs no communication costs. Queries are flooded to all nodes at a cost of O(n). Responses are sent back to the source of the query at a cost of O($\sqrt{n}$).

- Data-Centric Storage (DS): Here, after an event is detected the data is stored by name within the sensor network. The communication cost to store the event is O($\sqrt{n}$). Queries are directed to the node that stores events of that name, which returns a response, both at a cost of O($\sqrt{n}$)..

Our scenario is described by several parameters. [5] We consider a sensor network with n nodes equipped to detect T event types. We let Dtotal denoting the total number of events detected, Q denote the number of event types for which queries are issued, and Dq denote the number of events detected for the types of events queried for. We assume there is no more than one query for each event type, so there are Q queries in total.

2.2 Approximate Communication Costs

From the literature present in [5] The communication costs of the three methods can be approximated using the asymptotic costs for floods (O(n)) and direct routing (O($\sqrt{n}$)). Below we present approximations for both the total number of packets and

the packets arriving at the access point. We assume that the packet count at the access point is a good estimate of the hotspot usage, since we expect that the access point to be the most heavily used area of the sensor network.

- External Storage:
  a. Total: Dtotal$\sqrt{n}$
  b. Hotspot: Dtotal
- Local Storage:
  a. Total: Qn+Dq$\sqrt{n}$
  b. Hotspot: Q+Dq
- Data-Centric Storage:
  a. Total: Q$\sqrt{n}$+Dtotal$\sqrt{n}$+Dq$\sqrt{n}$ (list)
  b. Hotspot: Q+Dq (list) or 2Q (summary)

where (list) indicates a full listing of events is returned (requiring a packet for each event) and (summary) indicates only a summary of events is returned (requiring only one packet). These calculations support a few relatively obvious points. First, if all other parameters are held fixed, then as n gets large the local storage method incurs the highest total packet count. Second, external storage always incurs a lower total message count than data-centric storage, but the ratio $1+\frac{Q+Dq}{Dtotal}$ is unlikely to be large if there are many events detected (and, if there is at least one event detected of each type, this ratio is bounded by 3). Third, if Dq_Q and events are summarized, then data-centric storage has the lowest load (of all three methods) on the access path. Fourth, if events are listed and $Dtotal_Dq$ then data-centric storage and local storage have significantly lower access loads than external storage.

2.3 Summary

This suggests that data-centric storage is preferable in cases where (a) the sensor network is large, (b) there are many detected events and not all event types are queried, so that $Dtotal_max[Dq; Q]$. This performance advantage is further increased if summaries are used. However, if if the number of events is large compared to the system size, $Dtotal > Q\sqrt{n}$, and event lists (rather than summaries) are used, then local storage may be preferable.

Chapter 3

Overview of DHT

This chapter gives the overview of structure of DHT and routing techniques. DHT-based routing [4] is a new routing paradigm that has been proposed recently in the literature . The designed is inspired by DHTs overlay. DHTs are useful in WSNs because they can be used to implement scalable network services. The following subsections describe general model of DHTs-based routing.

3.1 The structure

In the model [5] as shown in fig 3.1, it uses random unsigned integers to identify nodes. Each node is assigned a random unique identifier from some namespace [0, 2n) and maintains pointers to its overlay-neighbors, where n is total number of nodes in physical network. In fig. 3.1, with overlay topology, A has two overlay neighbors, B and C. Each node needs to maintain a physical neighbor set that it can communicate with its Neighbors. An overlay neighbor for a node is chosen to be a node whose identifier satisfies some relationships with the former's identifier; this exact relationship is prescribed by the DHT. For this purpose, a pointer is maintained for every overlay neighbor as a source-route (SR) through the network. This SR is consisting of a set of physical links that establish a path from the node hosting the pointer to its overlay-neighbor. Therefore, in fig. 3.1, A uses the source-route D, E to reach

B; P1(A:DE:B), and the source-route F, G to reach C; P2(A:FG:C). P1 and P2 are periodically refreshed from the originating node, and if it detects that its overlay neighbor is not reachable because of the failure of an on-path link or node or the neighbor itself, it initiates a joining procedure once again. Note that all relationships are soft state and can thus adapt to changes.

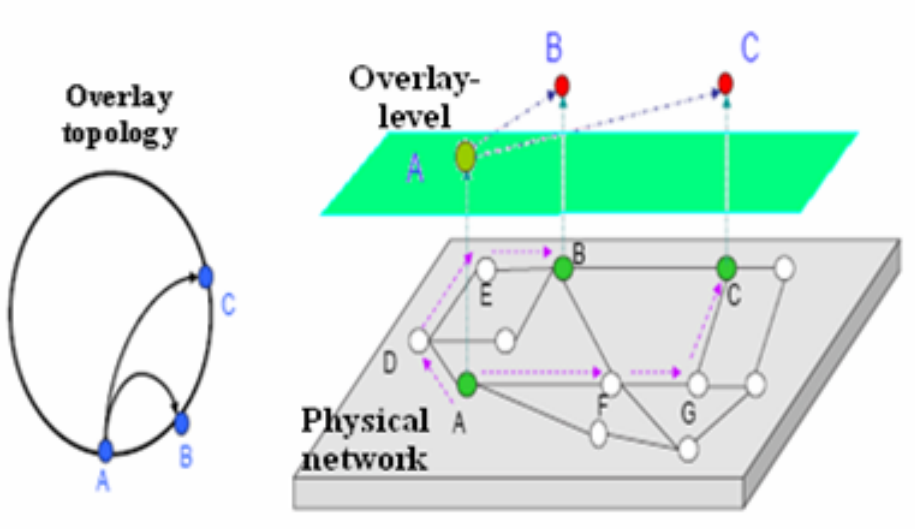

Figure 3.1: Model of DHT-based routing

3.2 Routing

The mechanism relies on greedy routing for routing to a destination. When the node need to send the packet to a destination, it will select the overlay neighbor that makes the most progress in the namespace, and then forwards the packet along the pointer to the DHT-neighbor. Forwarding along this pointer can be achieved either through a source route inserted by the sender or through embedded state in the network in the form of incremental source routes to the overlay neighbor. When the packet reaches the overlay neighbor, it repeats the same greedy routing process until the packet makes it all the way to the destination. Therefore, routing proceeds at two levels. First, along the overlay from one overlay neighbor to another. Second, from

one overlay neighbor to another along the pointer source route.

3.3 Dealing with Churn issues

When a new node arrives,[4] it discovers and constructs paths to its overlay-neighbors by routing using a physical neighbor that has already joined to act as a proxy. This is possible by the boot-strapping nature of DHTs. A node can simply route to its own identifier, and discover its overlay neighbors. When a new node routes to its new identifier, the path taken by this packet as a concatenation of all the pointers, its traverses constitutes a source-route to its prospective overlay neighbor. The new node can still use this in setting up a pointer to its neighbor. When a node leaves the network or a link fails, this event is needed to update in namespace. Therefore, all nodes had pointers to that node or traversing that link will soon realize the failure of the link, and will effectively rejoin the network.

3.4 Benefits

P2P overlay in WSNs or DHTs based routing in WSNs can be useful in several ways such as efficient data lookup, guaranties on lookup times, location independence, overlay application, and so on. As we know that the main purpose of WSNs is to collect and storage data about phenomena of interest. Thus, looking up or reporting data of interest in one of the main operations in WSNs, and optimizing this operation is a fundamental research problem. DHTs provide near optimum data lookup times that could help solve the efficient data lookup problem in WSNs. Additionally, DHTs based data lookup operation is the only core function of P2P overlay. Thus, it has been used to build a wide variety of applications such as file systems, event notification, content distribution, and so on.

Chapter 4

GPSR

This chapter describe benefits of GPSR algorithm. Greedy Perimeter Stateless Routing (GPSR) [7] is a novel routing protocol for wireless Sensor Networks that uses the positions of routers and a packet's destination to make packet forwarding decisions. GPSR makes greedy forwarding decisions using only information about a router's immediate neighbors in the network topology. When a packet reaches a region where greedy forwarding is impossible, the algorithm recovers by routing around the perimeter of the region. By keeping state only about the local topology, GPSR scales better in per-router state than shortest-path and ad-hoc routing protocols as the number of network destinations increases. Under mobility's frequent topology changes, GPSR can use local topology information to find correct new routes quickly. We describe the GPSR protocol, and use extensive simulation of mobile wireless networks to compare its performance with that of Dynamic Source Routing. Our simulations demonstrate GPSR's scalability on densely deployed wireless networks.

Greedy Perimeter Stateless Routing(GPSR) , a routing algorithm that uses geography to achieve small per-node routing state, small routing protocol message complexity, and extremely robust packet delivery on densely deployed wireless networks.

GPSR's benefits all stem from geographic routing's use of only immediate-neighbor

information in forwarding decisions. Routing protocols that rely on end-to-end state concerning the path between a forwarding router and a packet's destination, as do source-routed, DV, and LS algorithms, face a scaling challenge as network diameter in hops and mobility increase because the product of these two factors determines the rate that end-to-end paths change. Hierarchy and caching have proven successful in scaling these algorithms. Geography, as exemplified in GPSR, represents another powerful lever for scaling routing.

Chapter 5

Overview of DHT Base Routing over WSN

5.1 Geographic hash table (GHT)

The idea [1] of integrating routing with the DHTs was first proposed in the Geographical Hash Table of Sylvia Ratnasamy which combines the DHT idea with geographic naming and routing. In GHT there is a space of identifiers for nodes and keys. Unlike other DHTs, this space is not virtual, but physical. GHT hashes keys into geographic coordinates, and stores a (key, value) pair at the sensor node geographically nearest the hash of its key. The system replicates stored data locally to ensure persistence, when nodes fail. GHT is built atop GPSR a geographic routing system for multi-hop wireless networks. GHT uses two additional related concepts: the "home node" and the "home perimeter". The home node of a point is the node which is geographically closest to that point in space. The home perimeter of a point is the set of edges that encloses that point. Nodes on home perimeter will store the data

The GHT provides a similar interface as DHTs: Put(k; v) stores v (the observed data) according to the key k, the name of the data. v=Get(k) retrieves the value stored associated with key k There are 2 major aspects of GHT:

5.1.1 Data Dissemination and Storage:

To disseminate [2] data and store them in the network, GHT first map data to a geographic position according to the key of the data, and then, use GPSR to disseminate the data to the position. Note that the packet used to disseminate data does not target any node, but a geographic position, which means that no receiver ever sees the packet addressed to its own identifier. Thus, which nodes will consume this packet and how they consume it are two crucial problems in GHT. In fig. 5.1, it given an example. Node a call Put(k,v) to disseminate data to the network. GHT maps the data to coordination in the network, which is marked in fig. 5.1. Then, using GHT, the packet is forwarded to node d, where it enters the perimeter mode when no neighbor is closer to the destination than node d, and then traverses the home perimeter that consists of node d, node f,and node e. When the packet return to node d, node d knows that it is the closest one to the destination coordination, and becomes the home node and consumes this packet. Data retrieval is also achieved by a similar approach. Namely, for example node a call Get(k), the request packet will target the same location, node d, and will eventually arrive one of the nodes on the home perimeter. Then, this node will reply this request from node a.

5.1.2 Data Refreshment and Release:

To ensure the consistence of the data and handle the failure of nodes, three parameters are used: Th: the interval between which the home node generates a refresh message to ensure that the nodes on the home perimeter have the data stored. This packet will be treat as a put packet; Tl: The interval between which the nodes storing data generate a refresh message to ensure a home node exists. This packet will be treat as a put packet; Td: The live time of data. If data has not been refreshed inTd seconds,[2] the data will be released, where Td = 3Th and Tl = 2Th. Although GHT was inspired by Distributed Hash Table systems like Chord and CAN , but it would not appropriate to adopt the DHT routing algorithms for use WSNs. These

algorithms require nodes to be interconnected in a fairly Rigid manner. On the Internet, node neighbor relationships are at the logical level; the underlying IP routing system logically connects nodes that are not immediate physical neighbors. But in WSNs environment, nodes have the limited power energy, maintaining routing among all pairs of nodes is unfeasibly expensive. And because neighbors in the DHT logical identifier space may be topologically far apart, each logical hop within a DHT may cost energy for many packet transmissions.

5.1.3 Summary

DCS offers reduced total network load and hotspot network usage as compared with external storage and local storage. Our analysis reveals that these benefits occur on sensor nets comprised of large node populations, and where many events are detected, but not all event types are queried.

Still need to investigate Foremost among these is the effect of varying node density. To avoid hashing keys to points outside the sensor net, GHT requires Approximate knowledge of a sensor net's boundaries an open research problem is to devise scalable distributed algorithms to map these (possibly changing!) geographic boundaries.

5.2 Chord for sensor networks (CSN)

How to efficiently [1] locate the sensor nodes is a fundamental problem confronted by WSNs applications. DHTs algorithms are known that they are an internet peer to peer protocols which provide near-optimum data lookup times for queries made on networks of distributed nodes . Muneeb Ali et al.have proposed a novel approach, Chord for a sensor network (CSN) that uses the Chord's DHT over the sensor network. CSN is used to bound times for data lookup, in the order of O(logN) messages in an energy efficient manner.

Unlike other DHTs, Chord provides strong guarantees and its lookup function runs

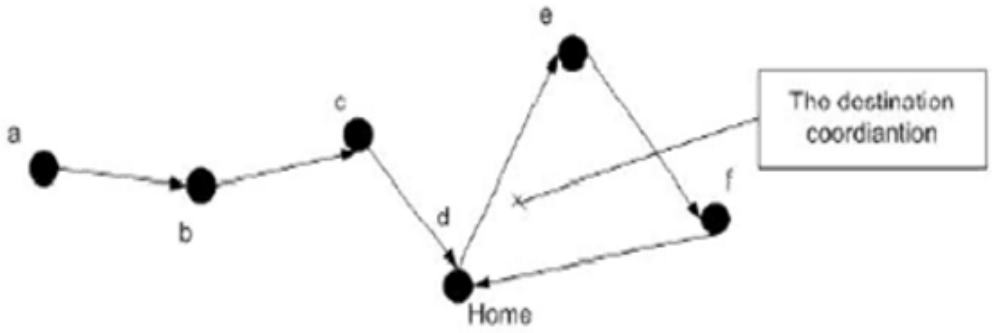

Figure 5.1: An Example Of GHT

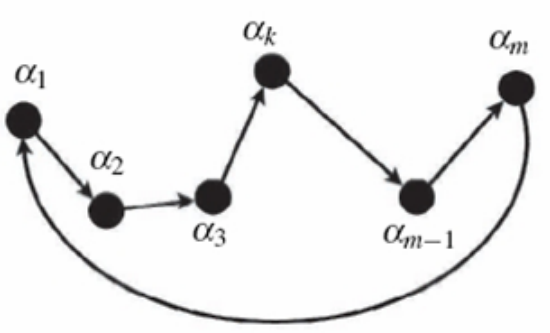

Figure 5.2: A1 and A2 m shouldn't Communicate Directly

in predictable time and always results in success or definite failure CSN runs on two modes that are the energy-efficient mode (EEmode) and the robust mode (Rmode). Because nodes in WSNs are limited energy, so each node needs to communicate with its nearest neighbor, and the logical ring of sensor nodes is geographically closest to it. To resolve ring problem, has used chain method and set-average method (SAM). In fig 5.2, Last sensor α m is making link with a first sensor α1. In fact, node αm has possibly larger geographical distance with α1, so that the communication among them would die. A solution can resolves this problem by using anti-clockwise in the ring. While operating in Rmode, CSN functions similar to Chord and can resolve lookups with a bound of O(logN) messages. But this efficiency in data lookup is achieved at a cost of greater energy consumption, as each node is directly sending messages to a node possibly farther away from it. The Rmode is used only if the application delay requirements can not be met by the EEmode. In the EEmode, the logical operation of sending a message to a node is separated from its actual routing and the routing of messages takes place by each node communicating only with its closest neighbor node. EEmode is the default mode of operation and CSN switches to the Rmode only if the application delay requirements can not be met by EEmode. Because, CSN uses data replication to handle node failures, but the sensor node is a limit device in term of storage and power, so the storage limitation can potentially become as a bottleneck for CSN network scalability.

5.2.1 Summary:

Unlike other DHTs, Chord provides strong guarantees and its lookup function runs in predictable time and always results in success or definite failure CSN runs on two modes that are the energy-efficient mode (EEmode) and the robust mode (Rmode).CSN uses data replication to handle node failures, but the sensor node is a limit device in term of storage and power, so the storage limitation can potentially become as a bottleneck for CSN network scalability

5.3 Virtual Ring Routing (VRR)

Virtual Ring Routing (VRR) [1]is a new network routing, is inspired by DHTs . It is implemented directly on top of the link layer. VRR occupies a unique point in the design space. On the other hand, VRR provides both traditional point-to-point network routing and DHT routing to the node responsible for a hash table key. In fig. 5.3, it gives relationship between the virtual ring and the physical network topology. VRR organizes the nodes into a virtual ring. Each node is assigned a random unique identifier. In virtual ring, all identifiers are ordered increasing identifier. Each node maintains both a virtual neighbor set (or vset) of cardinality r, and a physical neighbor set (or pset). Node identifiers are fixed, unique and location independent.

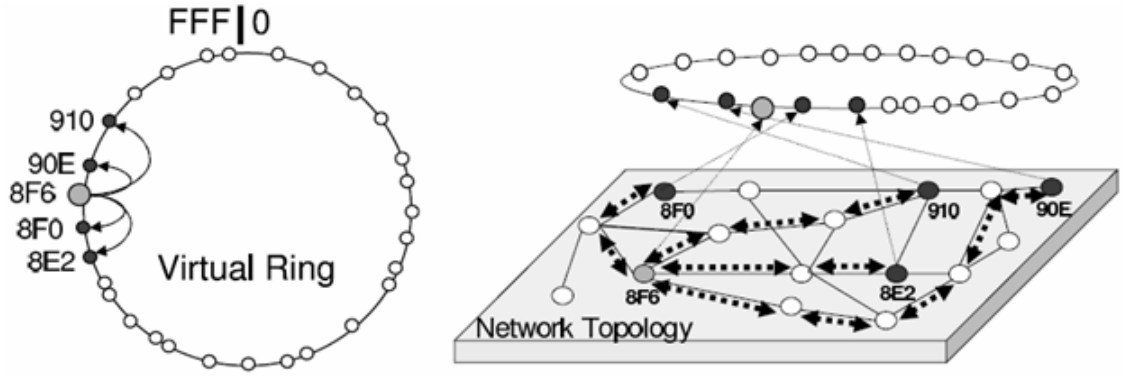

Figure 5.3: Relationship between the virtual ring and the physical network topology

In fig. 5.3[Left], it shows an example virtual ring with a 12-bit identifier space (with identifiers in base 16). It also shows the vset of the node with identifier 8F6 with r = 4. In fig. 5.3[Right], it shows the mapping between the virtual ring and the physical network topology and it shows the vsetpaths between node 8F6 and its virtual neighbors. All vsetpahts are stored routing paths between a node and each of its virtual neighbors. These are set up and maintained by VRR. In most cases, vsetpaths are multi-hop. They are also bidirectional because membership in the vset is symmetrical. Each node maintains a routing table with information about the vset-paths. A routing table entry identifies the two vset-path endpoints and the next

hop towards each endpoint. This information is maintained proactively. If each node maintains r vsetpaths to its virtual neighbors and the average path length is p, the total number of routing table entries in an n node network is nrp. Therefore, each node will have on average rp entries for vset-paths in its routing table: r entries for the paths to its virtual neighbors and r(p¿1) additional entries for vset-paths through the node. If we assume that these additional vset-paths end at nodes that are selected randomly and uniformly, the probability that a random node has a path to a random destination is O(rp/n). Therefore, a packet is expected to reach a node that has a vset-path to the destination after visiting O(n/(rp)) nodes, which will add only a constant stretch if p grows with pn. VRR does not use flooding at all and it uses only location independent identifiers to route. All control and data packets are routed as described above without any translation to location based addresses. In particular, control messages to setup new vset-paths are routed using the existing vsetpaths.

5.3.1 Summary:

VRR has three operations: forwarding, node joins, node and link failures. VRR has been designed to provide consistent routing to the node responsible for a key with high probability even with node mobility and failures. It uses local failure detection and a guaranteed failure notification mechanism to detect node and link failures quickly and with low overhead. Additionally, VRR does not need to transfer hash table data across nodes when they move because node identifiers are fixed and independent of the topology.

5.4 Topology based Distributed Hash Table (T-DHT)

In this approach,[6] Olaf Landsiedel et al have introduced the T-DHT as an infrastructure for data-centric storage, information processing, and routing in ad-hoc and

WSNs. It is a scalable infrastructure for data-centric storage in ad-hoc and WSNs. It does not rely on location information and works even in the presence of voids in the network. This method built a T-DHT by two steps: Step 1: The construction of a virtual coordinate space, which represents the network topology. Step 2: On top of these coordinates we build a two-dimensional hash-table, where each node maintains the data close to its virtual position. In this method, they have constructed a two-dimensional distributed hash-table on top of the virtual coordinate system. The coordinate space is divided among the participating nodes; each node maintains a rectangular area around its position in the virtual coordinate system. Each node maintains a routing table Containing the path to its neighbors in the coordinate system and the area each maintains. Routing in the 2-dimensional hash-space is intuitively, using its routing table a node forwards a message to its destination by simple greedy forwarding. Additionally, a node may have links to nodes, which are not Direct neighbors in the hash-table. Thus, it can use these to "short-cut" the route. Joining the T-DHT consists of three steps: Step 1: To join, a node must first find a node, which is already in the T-DHT. Step 2: Via the T-DHT routing it finds the node maintaining the zone of its position in the virtual coordinate system. This zone is equally spilt between the two nodes. Step 3: The new member informs its neighbors about its presence. In this approach, the distributed hash table offers a low routing overhead compared to the shortest path routing, and it uses a large quantity of energy for communication.

5.5 Cell Hash Routing (CHR)

CHR has [8] focused on the problem of implementing a distributed hash table (DHT) in wireless ad hoc networks. It is designed from scratch to cope with problems like limited available energy,communication range or node mobility. CHR supposed to overcome the limited available energy, communication range and node mobility by using position information on top of clusters of DHTs. CHR has divided the space

into equally size cells that the shape of squares, as the grid illustrated in fig. 5. All nodes inside the same cell belong to the same cluster. CHR has used the notion of cluster as a kind of "super node", where interactions occur at the level of clusters. Since cells are immutable, clusters can receive an dentification that does not change with time. Therefore, nodes can always determine the identification of their cluster, as well as identify other nodes in the same cluster, even in dynamic scenarios, where they move around and change from cell to cell. The size of the cells is limited by the communication range of the nodes, because a node in a cell must always listen to any a) b) other node either in its own cell or in any adjacent cell. Assumption that nodes have a communication range of R, so the square size is at most R/ 8 . CHR uses a routing scheme based on GPSR combined with a pre-processing algorithm. The hash function determines the single cluster that will hold the (key, value) pair. In the case of a given pair (keyA, valueA), the cluster whose identification equals hash(keyA) will be responsible for storing valueA. For instance, consider the ("Bob", 18) pair, where the key "Bob" hashes to 144. In this case, the value 18 should be stored at the cell 144. Formula (1) following shows how to determine the address of a cell in this scheme. Dx and Dy are the size of the space in the two dimensions, dx and dy are the sizes of each cell and Lx and Ly are the coordinates of the center point of the cell (it can also be any other point inside the cell). For instance, this equation is useful to let a node determine the number of its own cell.

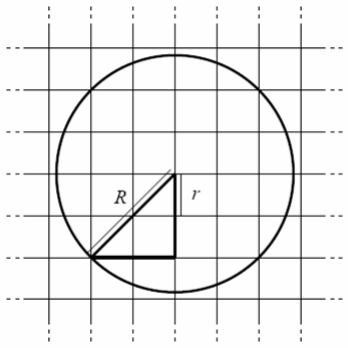

Figure 5.4: Division of the space into cells of fixed size

By using GPSR to route messages, CHR can follow an approach similar to GHT to resolve empty cells is that some keys may be left without nodes to store them. CHR is more robust, because a single key may be stored at many nodes, allowing the DHT to resist better to abrupt departure of nodes.

5.6 Comparison:

With analysis of the above methods, table 1 shows a comparison of them regarding to three main parameters in WSNs.

Approch	Scalability	Energy Efficent	Data Lookup Efficiency
GHT	+	+	++
CSN	+	+	++
T-DHT	++	–	+
CHR	+	+	+
VRR	+	+	++

Chapter 6

Problem,Solution and Challanges

6.1 Problem Statement:

- **Void cell:**[8]it may be possible cell doesn't have any node active in it. So when DHT maps key and value to void cell. It creates problems. And also suppose cell has only 1 or 2 nodes active then it may cause data unreachable when these nodes are going down.

- **General Non-Udg model [8]:** in more general models, it is possible for a node not to see some of its neighbors in its cell or in some adjacent cell.

- **Incorrect Determination of position:[8]** in all the papers, we assumed that nodes can always determine their position exactly, which is actually not the case. The consequence of this is that a node may consider itself to be in the wrong cell. In a sense, this resumes to the non-UDG model.

- **Cluster Induce Disconnection:** [8]occasionally, the clustering mechanism may disconnect the network. This event should be rare, especially in denser networks, where use of CHR is more worthwhile.

- **Fault tolerance requirements:**[8] one of the occurrences that CHR should try to avoid as much as possible is the loss of stored (key, value) pairs.

- **Network Life Time:** maximum network life time.

6.2 Possible Solution:

- **Void Cell:**When the number of nodes in a cell drops below l, the cell is considered empty. On the contrary, the cell needs to acquire h nodes before it is considered populated (h > l). Note that the value h should be fairly small. As a consequence, knowing h does not require much memory, because, in general, a node will not need to know all the neighbors in its own cell A cell leaving the network delivers its keys to its home cell. An entering cell needs to query its home perimeter to receive its keys. Additionally, it will also receive keys of empty cells for which it becomes the home cell.

- **Network Life Time:** Increase the network life time by implementing round robin algorithm between three paths between source and destination.

- **Fault Tolerance:** We can also use multiple paths to provide Fault Tolerance because if one path is fail to transmit packet than we have another path to transmit that packet.

6.3 Challenges for Scalable, Robust DCS

The challenge in any design for a DCS system is to meet scalability and robustness criteria despite the systems fundamentally distributed nature. Sensor nets represent a particularly challenging environment for a distributed storage system:

- **Node failures** may be routine; exhaustion of battery power and permanent or transient failure in a harsh environment are problems in any realistic sensor net deployment.

- **Topology changes** will be more frequent than on traditional wired networks. Node failures, node mobility, and received signal strength variations in real radio

deployments each independently cause neighbor relationships among nodes to change over time.

- **System scale** in nodes may be very great. Sensor nodes may be deployed extremely densely (consider the limit case of smart dust), and may be deployed over a very wide physical region, such that the total number of devices participating in the DCS system may be on the order of 106 or more nodes.

- **Energy constraints** will often be severe; nodes will operate from battery power. These challenges suggest several specific, important design criteria for ensuring scalability and robustness in the distributed storage system we envision:

  a. **Consistency:** a query for k must be routed correctly to a node Where(k,v) pairs are currently stored; if this node changes (e.g., to maintain persistence after a node failure), queries and stored data must choose a new node consistently.

  b. **Scaling in database size:** as the number of (k,v) pairs stored in the system increases, whether for the same or different ks, storage should not concentrate at any one node.

  c. **Scaling in node count:** as the number of nodes in the system increases, the system's total storage capacity should increase, and the communication cost of the system should not grow unduly. Nor should any node become a concentration point of communication.

  d. **Topological generality:** the system should work well on a broad range of network topologies.

  e. **Persistence:** a(k,v)pair stored in the system must remain available to queriers, despite sensor node failures and changes in the sensor network topology.

Chapter 7

Study of Network Simulator: NS-2

NS-2 is an open source system that is developed using C++ and Tool Control Language TCL. Researchers can freely add new components to the system to serve their own purposes. It provides support for IP protocols suite and many standard routing protocols for wire and wireless networks. There are unicast and multicast routing protocols for wire network and DSR, DSDV, AODV for wireless networks.

The network simulator (NS), which is a discrete event simulator for networks, is a simulated program developed by VINT (Virtual InterNetwork Testbed) project group (A Collaboration among USC/ISI, Xerox PARC, LBNL, and UCB). It supports simulations of TCP and UDP, some of MAC layer protocols, various routing and multicast protocols over both wired and wireless network etc. The basic structure of NS-2 is as shown in figure.

We can add new components to the system to serve their own purposes. It provides support for IP protocols suite and many standard routing protocols for wire and wireless networks. There are unicast and multicast routing protocols for wire network and DSR, DSDV, AODV for wireless networks.

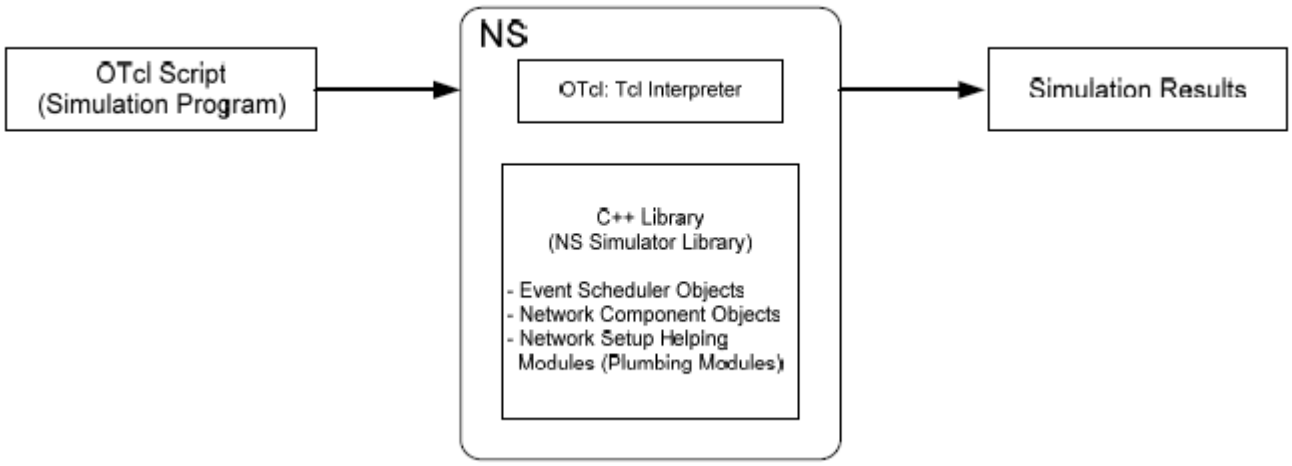

Figure 7.1: Basic Structure Of Ns-2

The network simulator (NS), which is a discrete event simulator for networks, is a simulated program developed by VINT (Virtual InterNetwork Testbed) project group (A Collaboration among USC/ISI, Xerox PARC, LBNL, and UCB). It supports simulations of TCP and UDP, some of MAC layer protocols, various routing and multicast protocols over both wired and wireless network etc. The basic structure of NS-2 is as shown in figure 7.1.

Basic structure of NS: To setup and run a simulation, a user writes an OTcl script, which is a simulation program to initiate an event scheduler, set up the network topology using the network objects and plumbing functions in the library, and to tell traffic sources when to start and stop transmitting packets through the event scheduler. When NS-2 which works as OTcl interpreter receives the OTcl script, it will set environment parameters following the received script. If a user wants to make a new network object, it will be easy to make a compound object from the object library, and plumb the data path through the object rather than write a new one. When the simulation is finished, the simulation results are produced in one or more text-based output files that contain detailed simulation data, which can be used to analyze directly.

Chapter 8

Implementation

8.1 Work Flow

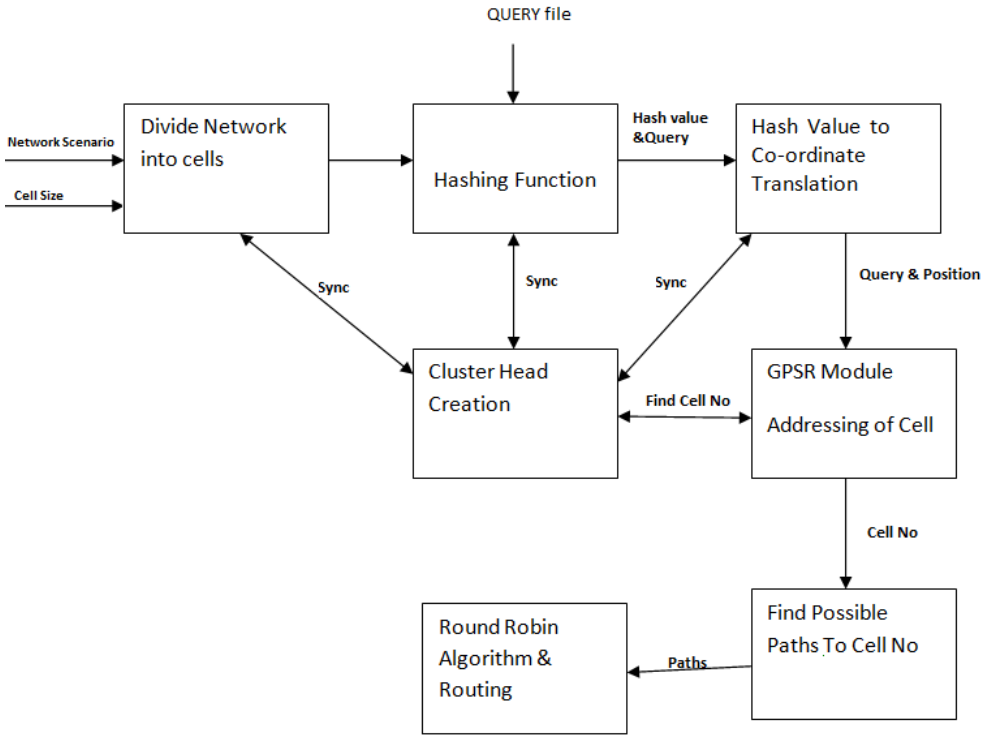

Figure 8.1: Work Flow

Work flow of implementation is shown as above.Here first we are dividing network into cells from the given network scenario as well as simultaneously creating cluster head for each cell.This module deal with managing cluster head.Now every query passed through hash function in each node which it passed through.Hash function will give node no to which query has to sent. As GPSR is dealing with geographical location therefore we need to translate hash value to co-ordinate value.After translating it to co-ordinate value,co-ordinates and query are to be sent to the gpsr module.GPSR module find the cell no and it cluster head from the co-ordinates value. and route the query to cell no.but in between gpsr finds all the possible paths from source to destination and Round robin algorithm forward the query packets to each path.

8.2 Simulation of GPSR and GAF protocol using NS-2

8.2.1 Installing NS-2.33 and Adding GPSR module on Fedora 8

a. Prepare neccesary files for installation

   1 NS-2.33 package: ns-allinone-2.33.tar.gz

   2 Download gpsr.cc,gpsr.h,gpsr.o

b. Download NS-2.33 and install it

   1 wget http://www.isi.edu/nsnam/dist/ns-allinone-2.33.tar.gz OR wget http://www.intern
   allinone-2.33.tar.gz

   2 tar zxvf ns-allinone-2.33.tar.gz

   3 cd ns-allinone-2.33/

   4 ./install

c. Download GPSR module and add to ns-2.33

[1].Copied gpsr.cc and gpsr.h and we pasted them under ns-allin-one/ns-2.33/GPSR.

[2].Defined GPSR packet type in packet.h

```
    enum packet_t
    ....
    PT_GPSR,
// insert new packet types here
    PT_NTYPE // This MUST be the LAST one
    };
   class p_info {
   public:
   p_info() {
   ....
   name_[PT_GPSR]= "GPSR";
   name_[PT_NTYPE]= "undefined";
   }
   static bool data_packet(packet_t type) {
   return (
   ....
   (type) == PT_HDLC || \
   (type) == PT_GPSR \
   );
#define DATA_PACKET(type) (
....
(type) == PT_SCTP_APP1 || \
(type) == PT_GPSR \
)
```

[3].Defined default value of GPSR agent in ns-default.tcl

```
Agent/GPSR set packet_size 51
```

[4].Defined new packet type in ns-packet.tcl.

```
foreach prot {
....
# Other:
....
GPSR
}
```

[5].Wrote "gpsr.o" in Makefile

```
GPSR/gpsr.o
```

[6].Went to the directory where is the ns2, run "make clean", "make", "make install" commands to configure the ns-2.33

Figure 8.2: Download GPSR module and add to ns-2.33

d. Set the environment variables to make NS-2.33 works

```
1. cd ~
2. gedit .bashrc
        #LD_LIBRARY_PATH
        OTCL_LIB=~/ns-allinone-2.33/otcl-1.8
        NS2_LIB=~/ns-allinone-2.33/lib
        X11_LIB=/usr/X11R6/lib
        USR_LOCAL_LIB=/usr/local/lib
        export
        LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR
        _LOCAL_LIB
        #TCL_LIBRARY
        TCL_LIB=~/ns-allinone-2.33/tcl8.4.5/library
        USR_LIB=/usr/lib
        export TCL_LIBRARY=$TCL_LIB:$USR_LIB
        #PATH
        PATH=$PATH:~/ns-allinone-2.33/bin:~/ns-allinone-2.33/tcl8.4.5/unix:~/ns-allinone-
        2.33/tk8.4.5/unix
3. source .bashrc
```

Figure 8.3: Set the environment variables to make NS-2.33 works

e. Test and debug

```
1 ./test
```

8.3 Building Cell Hash Routing Protocol in NS2.

8.3.1 Cell-Division Implemetation:

We are using GAF(geographic Adaptive Fidelity) protocol for topology control and dividing topology into Cells.the idea is to divide the area into rectangles that are small enough such that any node in one rectangle can communicate with any other node in adjacent rectangle.Node should commuticate at maximum radiorange. Figure illustrates this relationship for rectangle length r and maximum radio range R.

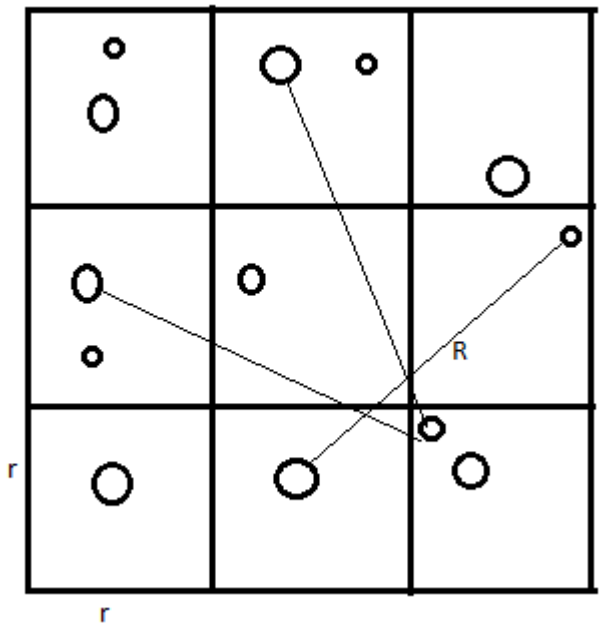

Figure 8.4: GAF

The distance between two such critical nodes is $\sqrt{r^2 + (2r)^2}$. As this distance has to be smaller than R it follows that r < R$\diagup\sqrt{5}$.since node are assumed to know their location,they can easily construct such equivalency rectangles . We have integrated GAF module with NS-2 to use it in our TCL script.

8.3.2 Energy model implementation:

- Total Energy: 80
- Transmission Power: 1.0
- Receiving Power: 1.0
- Sleep Mode Power: 0.05

8.3.3 Node Configuration:

```
puts "Configuring Nodes ($val(nn))"
$ns_ node-config -adhocRouting $val(adhocRouting) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channel $chanl \
       -topoInstance $topo \
            -wiredRouting OFF \
       -mobileIP OFF \
       -agentTrace $val(agttrc) \
            -routerTrace $val(rtrtrc) \
            -macTrace $val(mactrc) \
            -movementTrace $val(movtrc)\
            -energyModel $opt(energymodel) \
            -idlePower 1.0 \
            -rxPower 1.0 \
            -txPower 1.0 \
            -sleepPower 0.001 \
            -transitionPower 0.2 \
            -transitionTime 0.005 \
            -initialEnergy $opt(initialenergy)
$ns_ set WirelessNewTrace_ ON
```

Figure 8.5: Node Configuration

8.3.4 Creating Nodes and Topology Control using GAF:

```
for {set i 0} {$i < $val(nn)  } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 0;# disable random motion
                $node_($i) attach-gafpartner
                $node_($i) unset-gafpartner ;#
        }

        $ns_ node-config -initialEnergy $opt(initialenergy)

             # The rest node is GAF node

                for {set i 10} {$i < $val(nn) } {incr i} {

                    set node_($i) [$ns_ node]
                    $node_($i) random-motion 0

                #attach gaf agent to this node, attach at port 254
                    set gafagent_ [new Agent/GAF [$node($i) id]]
                    $node_($i) attach $gafagent_ 254
                    $node_($i) attach-gafpartner
                    $gafagent_ adapt-mobility $opt(gaf4)
                    $ns_ at 0.0 "$gafagent_ start-gaf"
                }
```

Figure 8.6: Creating Nodes and Topology Control using GAF

8.3.5 DHT implementation:

Basic Mechanism:

In the most basic setting, the hash function determines the single cluster that will hold the (key, value) pair. In the case of a given pair $keyA, valueA$, the cluster whose identification equals hash(keyA) will be responsible for storing valueA.

Hashing Function:

Now we need to build hash function that maps the event name to some node. For that we need hash function that gives integer value from the input of string and hash key must be in range of no of nodes in networks. For that purpose we are using following algorithm.

- H: Hash key(output of hash function)
- E: Event name
- R: Hash Size(No of nodes in networks)

```
Int Hash(char *E,int R)
{
For (H=0;*E!=NULL;E++);
H=*E+(31*H);
Return H % R;
}
```

Addressing of the Cells:

We illustrate one possible mapping with a very simple example that assumes a bounded geographical space whose bounds are known by all the nodes. Equation 1 shows how to determine the address of a cell in this scheme. Dx and Dy are the size of the space in the two dimensions, dx and dy are the sizes of each cell and Lx and Ly are the coordinates of the center point of the cell. For instance, this equation is useful to let a node determine the number of its own cell.

$$A = [Dx/dx] * [Ly/dy] + [Lx/dx] \tag{8.1}$$

The reverse correspondence is also useful to allow nodes to perform geographical routing in G. Equations 2 and 3 determine the center point (Lx;Ly) of the cell. c represents the number of columns and is computed as c = [Lx/dx] while % is the

remainder of the division. To route to a given cell A, nodes need to determine the center point (Lx;Ly) of the destination cell, before they apply the GPSR routing algorithm. These equations are needed because geographical routing takes place using the center points of the cells (graph G), while the DHT addresses of the cells are only logical. Consider again the "Bob", 18 pair, where the key hashes to 144. To compute the center (Lx;Ly) of the target cell, a given node would have to replace A by 144 in the Equations 2 and 3.

$$Ly = dy([A/c] + 0.5) \tag{8.2}$$

$$Lx = dx([A\%c] + 0.5) \tag{8.3}$$

8.4 Implementing Round Robin Algorithm

First of all when any packets sequence for the same destination comes to the node,node finds all the neighbor cells and its corresponding cluster head.After it path finding algorithm finds the appropriate three paths in the direction of destination as shown in figure 8.7.

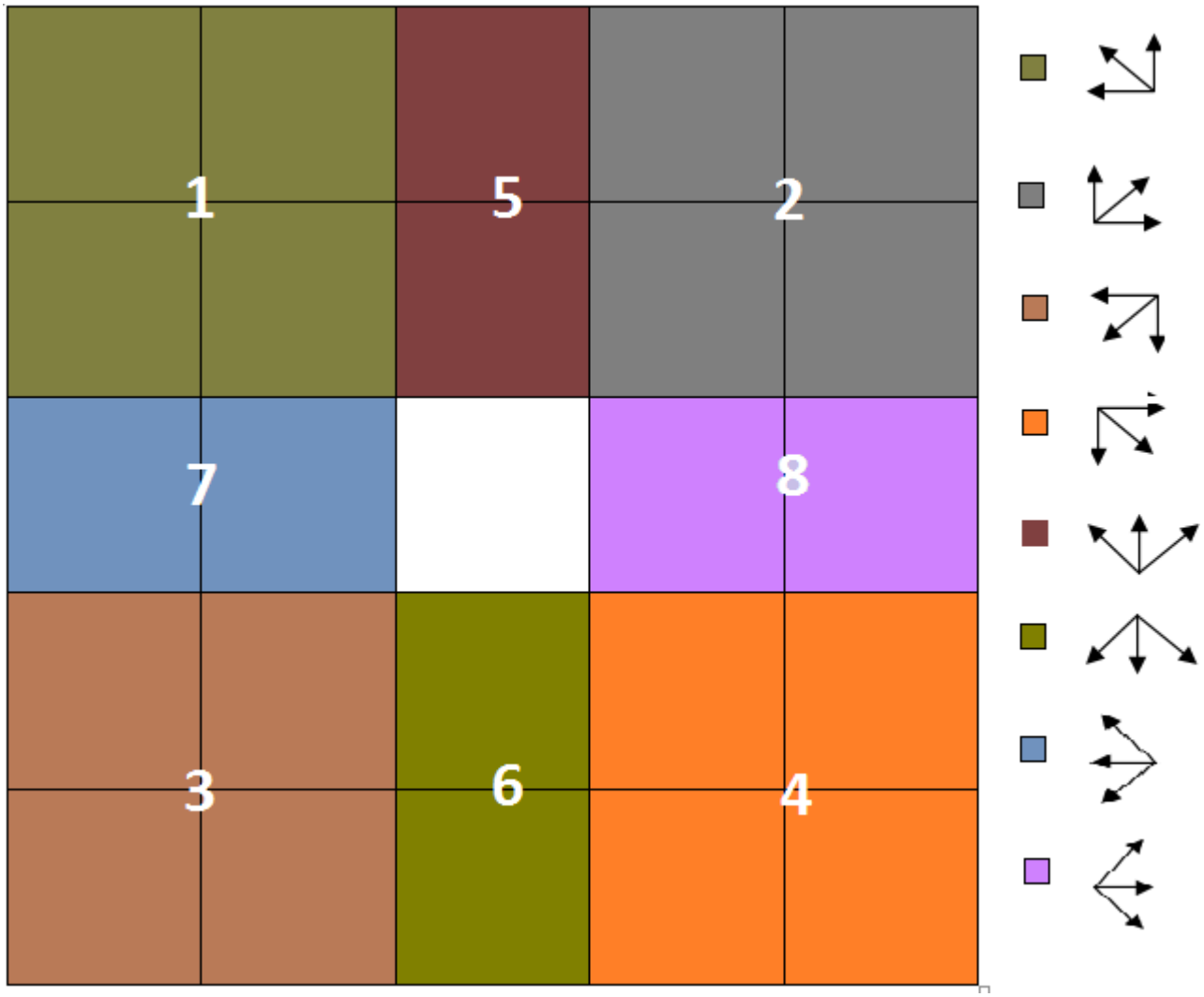

Figure 8.7: Multiple Path Finding Algo

In figure 8.7 we have shown 8 different regions and also we have shown directions of paths to the destination node of that particular region. Now we have three different paths to each destination node and we are doing round robin over these three path for sending packets from one source to destination.

Chapter 9

Results

9.1 Comparison Of Four Successive Implementations

Here we have shown comparison of Simple GPSR routing(implementation-1),Grided network routing(implementation-2) , Cell hash routing(implementation-3)and Round Robin implementation in CHR(implementation-4). We used ns2 simulator for simulation of these three implementations for 50, 100 and 200 nodes.In simulation scenario we took 1000 $\times$ 1000 area and cell size we took 200. As by default radio range in ns2 is 250 so from Equation in section 8.2.1. We calculated maximum cell size is 200. In the other side we took initial energy of each node is 80. Rx and Tx power would be 1.0 and idol and sleep power would be 0.05. As we need to optimize network life time and as well as speed up routing method. We have taken three matric parameters for comparison Average Path Length(APL),End to End Delay(E2E Delay) and Network Life Time(NLT). Average Path Length is no of hops from which query passed through from source to destination. E2E Delay is time takes to pass data from source to destination and NLT is time at which half of the nodes are down. As well as we also need to check the overhead of hashing algorithm. We have also taken overhead consideration in our simulation.

Table I: Simulation Scenario

Area	1000 × 1000
Cell Size	200 × 200(r<R/$\sqrt{}$(5),R=250)
Initial Energy	80
Rx Power	1.0
Tx Power	1.0
Sleep Mode	0.05
Simulation Time	100 sec

Table II: E2E Delay(msec)

No of Nodes	Simple GPSR	GRID	CHR	RR
50	1.49	1.12	1.17	2.1
100	4.6	2.45	2.5	3.11
200	7.38	6.38	6.84	7.20

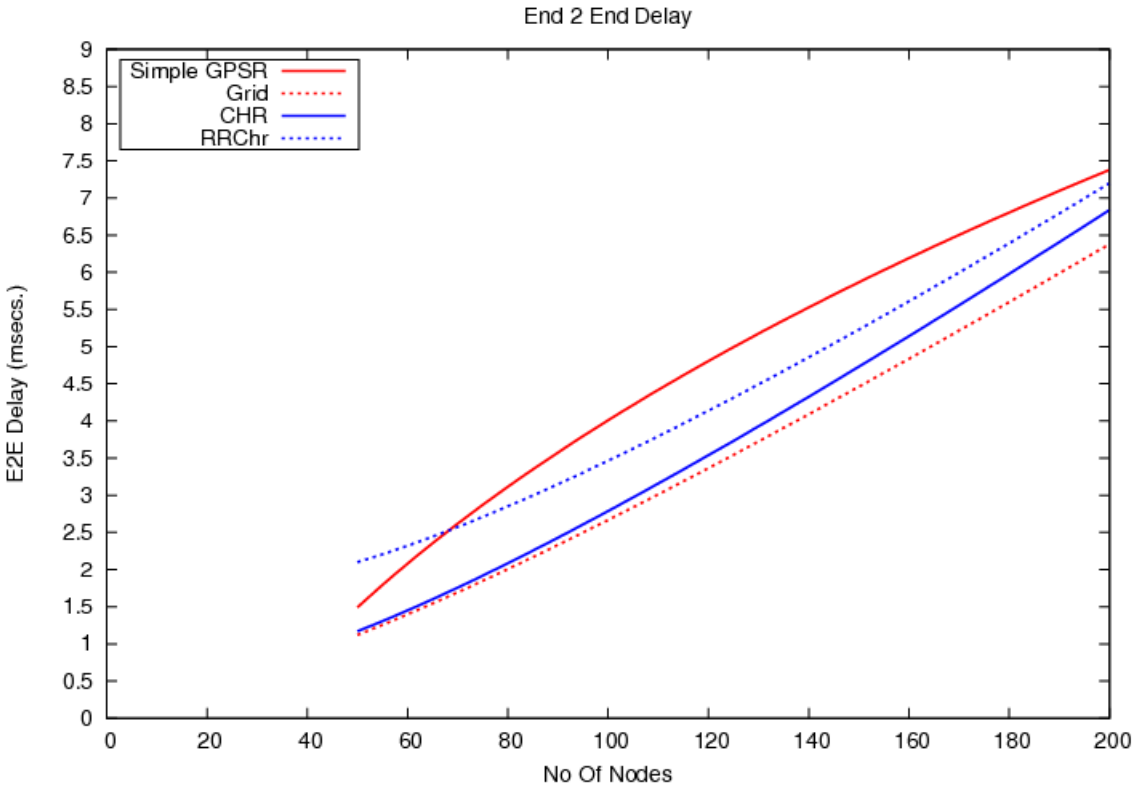

Figure 9.1: End To End Delay(msec))

Analysis: We can see from figure 9.1 that Clustering reduces End To End Delay but when we applied Hashing on the second implementation than hashing slightly increases End To End Delay because of Hashing Calculation of nodes. After this when we applied Round Robin Algorithm on third implementation than Round Robin Algorithm also increases End To End Delay because of Multiple Paths that all are not shortest paths from source to destination so packets have to travel more than in Pervious implementation. Overall last implementation improves End to End Delay by 2.43 %.

Table III: Hashing Overhead

No of Nodes	Overhead(msec)
50	0.04809
100	0.05096
200	0.45861

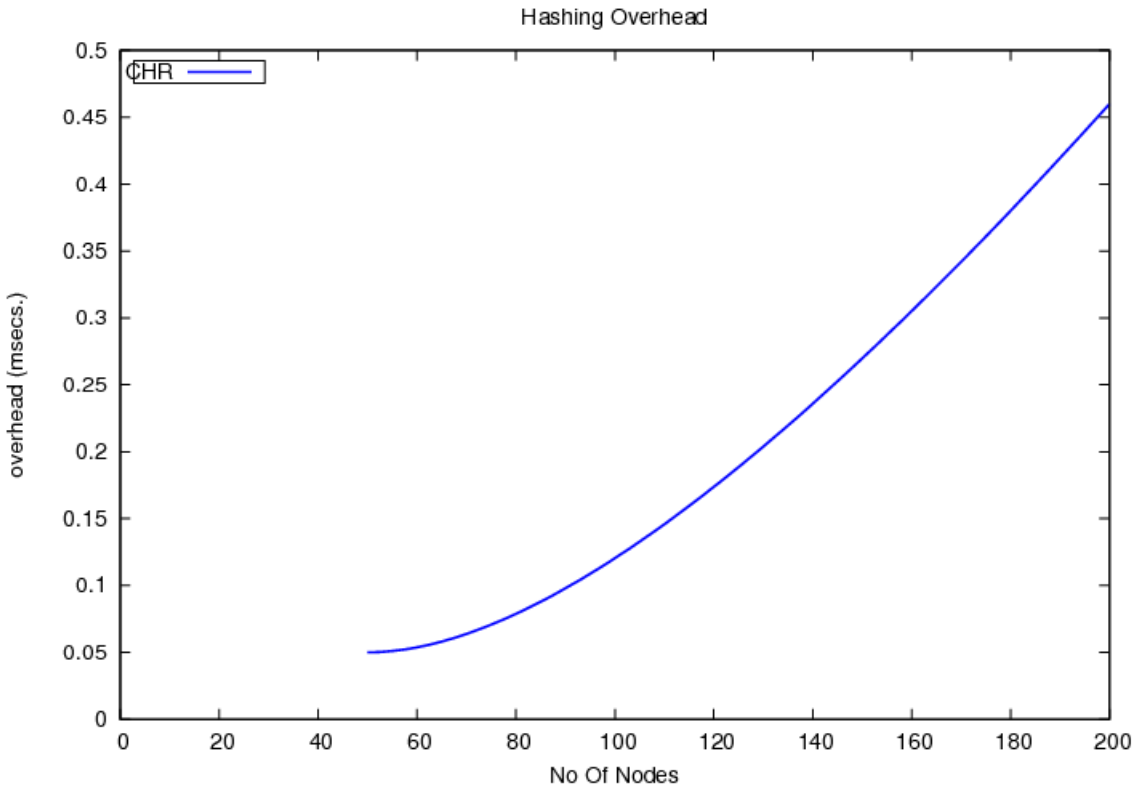

Figure 9.2: Overhead(msec)

Analysis: Figure 9.2 is showing the overhead of Hashing.Here we have taken increase in End To End Delay because of Hashing as overhead. It grows exponentially with large no of nodes in the network.

Table IV: Average Path Length

No of Nodes	Simple GPSR	GRID	CHR	RR
50	8.36	7.66	7.03	8.01
100	14.1	10.69	10.36	12.03
200	16.32	13.53	13.25	16.10

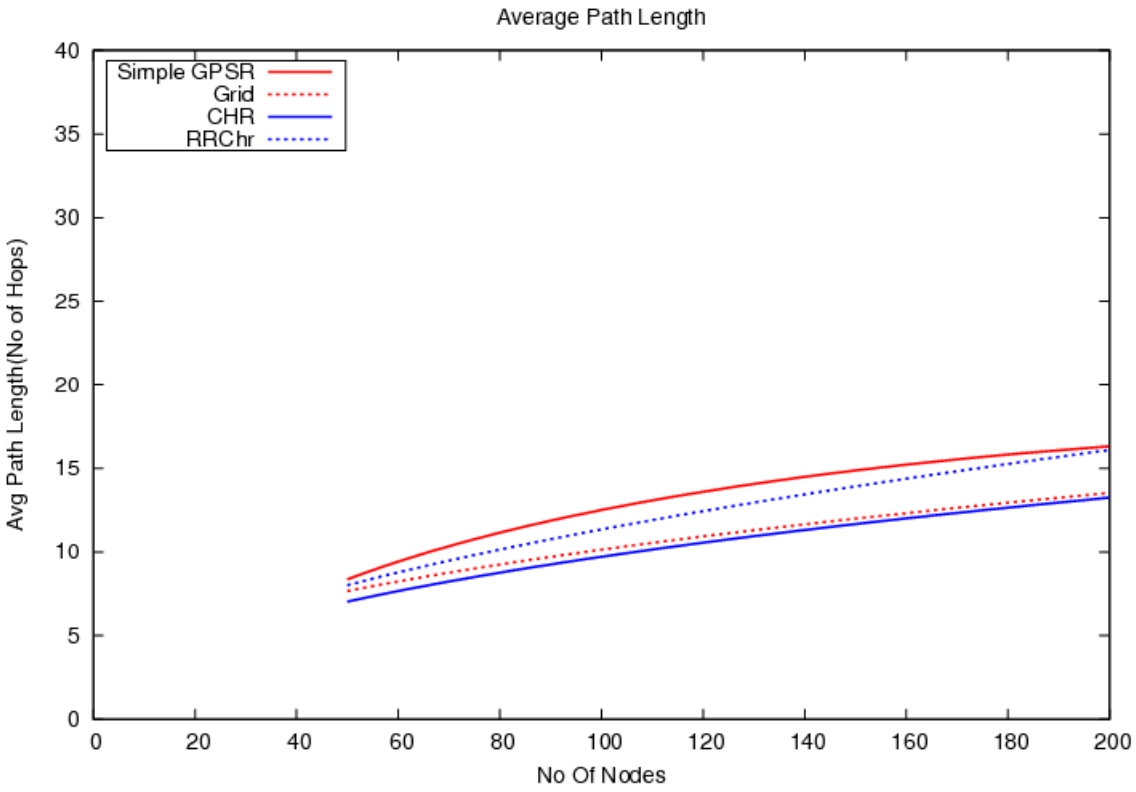

Figure 9.3: Average Path Length

Analysis: Figure 9.3 is showing the average no of hops that packet has to travel to reach destination. As we know clustering always reduces hop count so when we applied clustering on simple GPSR than it reduces average hop counts but when we applied Round Robin Multiple paths Algorithm than it slightly increases the hop counts because of multiple paths.Here proposed implementation improves Average Path Length by 1.4 % over simple GPSR.

Table V: Network Life Time(sec)

No of Nodes	Simple GPSR	GRID	CHR	RR
50	90.03	91.23	91.8	105.21
100	80.23	81.32	82.05	90.12
200	80.12	81.8	83.25	95.2

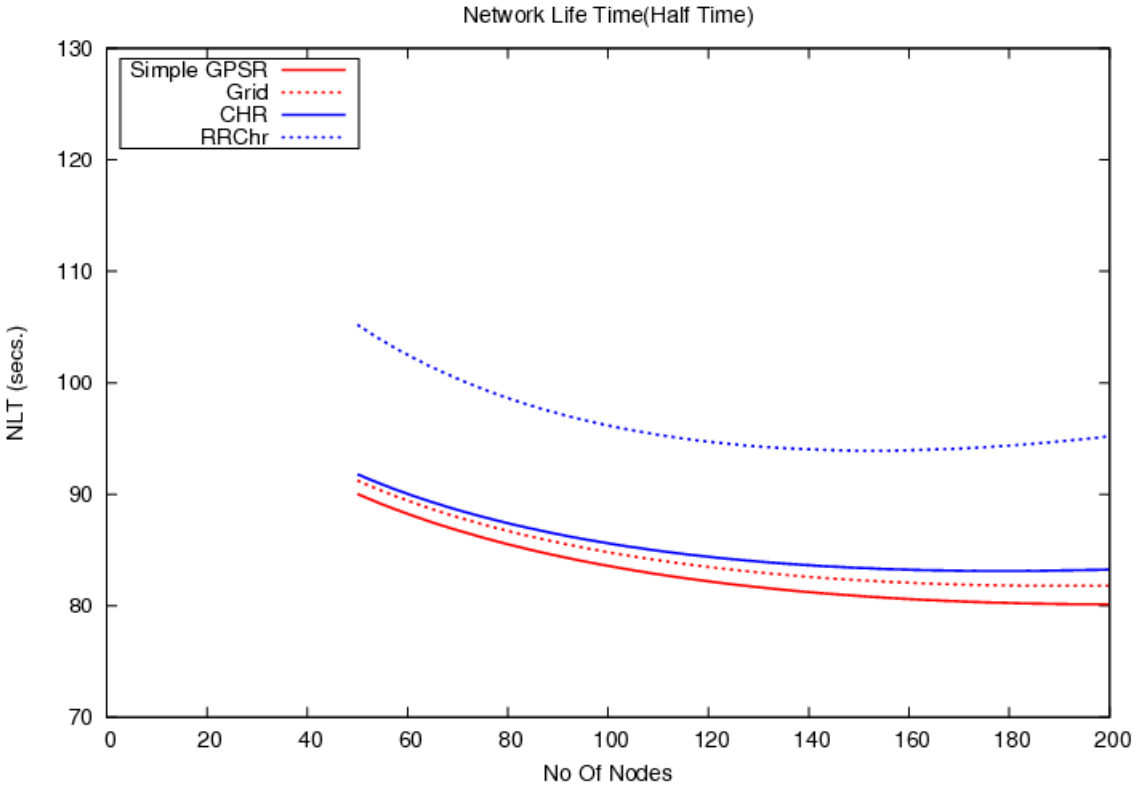

Figure 9.4: Network Life Time

Analysis: Here we have taken half time as time at which half of nodes will go down, As we can see from the figure 9.4 that Round Robin Algorithm increases the Half time of Network by around 18 %.

Table VI: Total Life Time(sec)

No of Nodes	Simple GPSR	GRID	CHR	RR
50	139.9	143.26	144.17	150.13
100	130.46	132.26	134.34	142.64
200	125.16	130.18	132.26	140.12

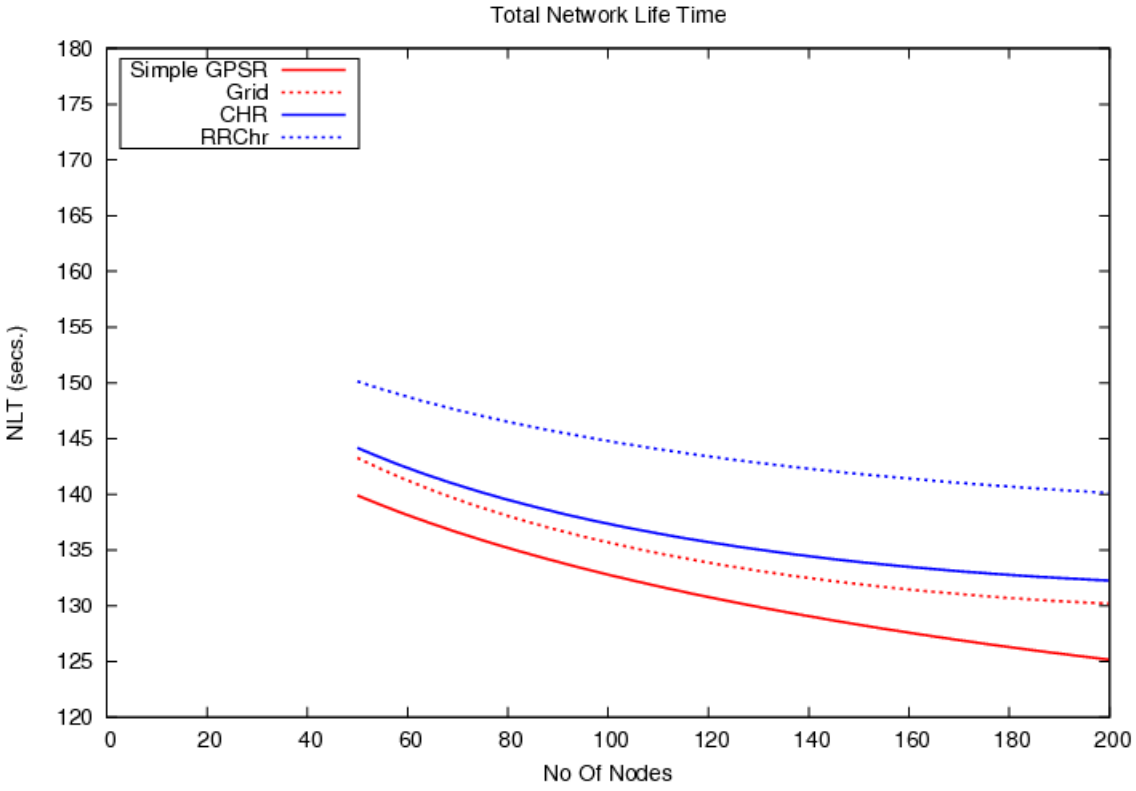

Figure 9.5: Total Life Time

Analysis: Here we have taken Total Life time as time at which all nodes will go down, As we can see from the figure 9.5 that Round Robin Algorithm also increases total time of Network by 15 %.

Table VII: Throughput

No of Nodes	Simple GPSR	GRID	CHR	RR
50	458.06	508.06	590.39	495.12
100	1698.94	1700.11	1919.4	1924.10
200	2120.23	2300.28	2500.16	2450.11

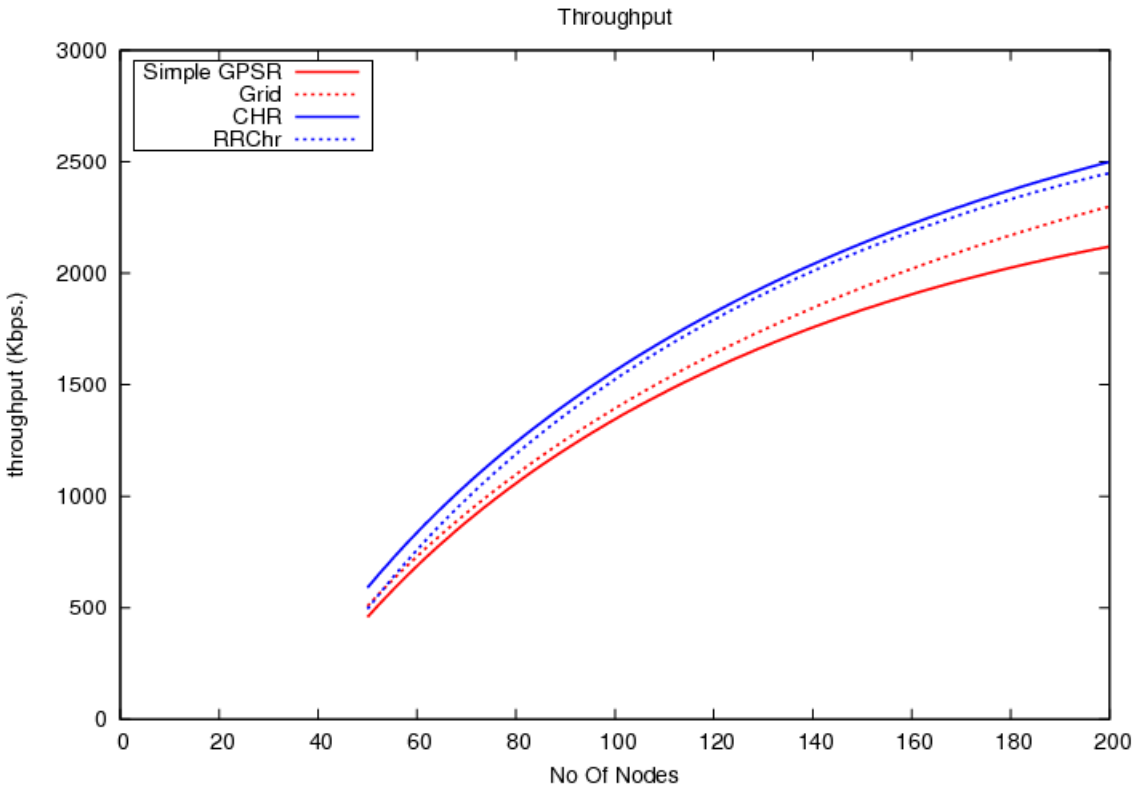

Figure 9.6: Throughput

Analysis: Throughput is bandwidth of channel which is increasing when we applied clustering and hashing on simple GPSR but when we applied Round Robin Over it than it is slightly lower then third implementation because of longer paths[figure 9.6]. Proposed implementation improves throughput from 10 to 15 %.

Table VIII: Energy

No of Nodes	Simple GPSR	GRID	CHR	RR
50	700	644.32	670.12	680.45
100	1200	1121.18	1140.23	1180.43
200	1600	1200.13	1326.16	1405.34

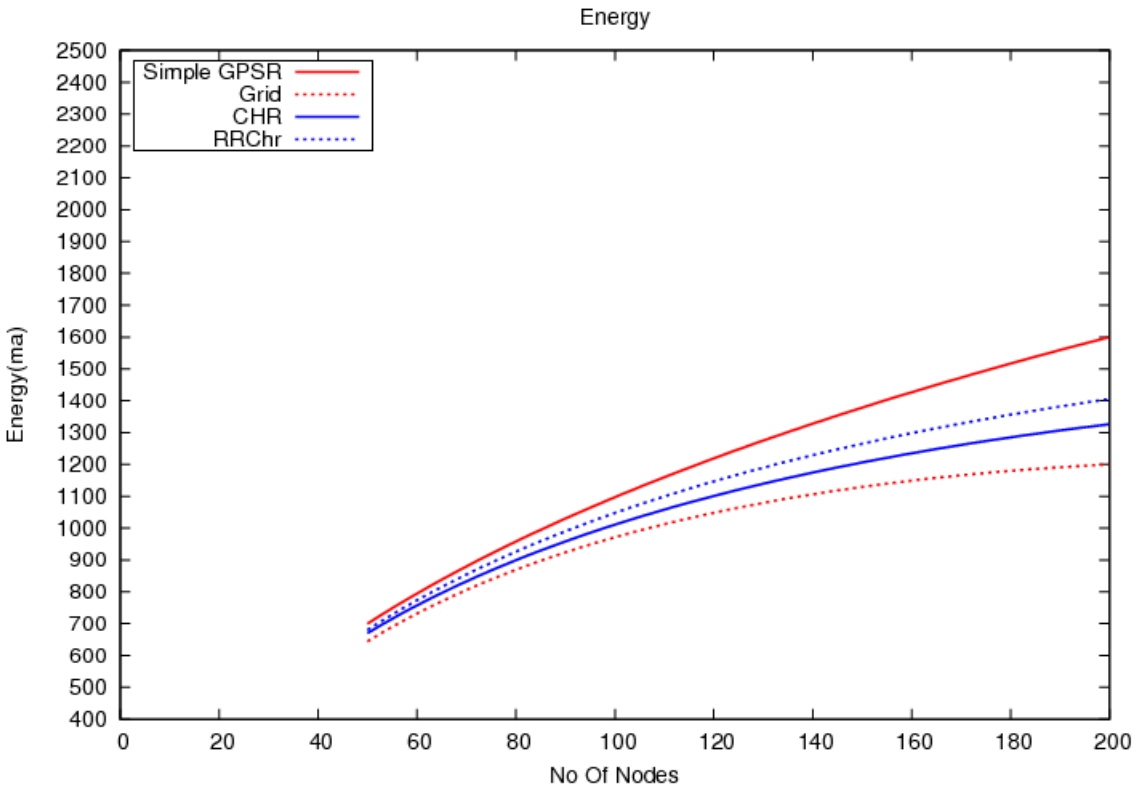

Figure 9.7: Energy

Analysis: Figure 9.7 is showing total energy consume in network during network simulation time of 100 seconds. Implementation increases total energy consumption around 5 % over CHR as we have a larger average path length. Because of Larger Average Path Length more nodes are sending and receiving same packet to reach destination in this way they consume more energy.

Chapter 10

Conclusion and Future Scope

10.1 Conclusion

In this dissertation, I have proposed a method to implement Round Robin Algorithm in CHR protocol for wireless sensor network with some assumptions. It finds the multiple paths from source to destination and apply round robin over those paths. So it reduces the load over single path and every path gets a fair chance to transmit packets. Proposed method increases network life time[figure 9.4 and 9.5].

From the results we can see that goal of thesis - "Distributed Hash Table For Scalable Wireless Sensor Network" is achieved but as an odd effect this Proposed method increases End To End Delay.

It is to be noted that throughput decreases slightly but it is almost close to CHR implementation from figure 9.6.

10.2 Future Scope

In future, the idea presented over here can be extended by solving the following problems by possible solution mentioned here,

Dynamic Cell Structure: When the number of nodes in a cell drops below l, the cell is considered empty. On the contrary, the cell needs to acquire h nodes before it is considered populated (h > l). Note that the value h should be fairly small. As a consequence, knowing h does not require much memory, because, in general, a node will not need to know all the neighbors in its own cell. A cell leaving the network delivers its keys to its home cell. An entering cell needs to query its home perimeter to receive its keys. Additionally, it will also receive keys of empty cells for which it becomes the home cell.

General non-UDG model: In more general models, it is possible for a node not to see some of its neighbors in its cell or in some adjacent cell. One possible approach to overcome this problem is to create routing tables to reach only the invisible nodes inside the cell or adjacent cells.

References

[1] Vinh Vu Thanh, Hung Nguyen Chan, Binh Pham Viet, Thanh Nguyen Huu "A survey of routing using DHTs over Wireless Sensor Networks" in The 6th International Conference on Information Technology and Applications (ICITA 2009)

[2] Sylvia Ratnasamy and Brad Karp ICIR/ICSI, Berkeley, CA 94704 sylviar, Yin Li and Fang Yu Berkeley EECS, Berkeley, CA 94704 yinli , Deborah Estrin UCLA Comp. Sci., LA, CA 90095,

[3] Ramesh Govindan USC Comp. Sci., LA, CA 90089,Scott Shenker ICIR/ICSI, Berkeley, CA 94704 "GHT: A Geographic Hash Table for DataCentric Storage"

[4] Olaf Landsiedel, Stefan Gotz, Klaus Wehrle Distributed Systems Group RWTH Aachen, Germany firstname.lastname@cs.rwth-aachen.de "Towards Scalable Mobility in Distributed Hash Tables" in Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)

[5] Frank Dabek "A Distributed Hash Table" in Submitted to the Department of Electrical Engineering and Computer Science in partial fullfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology September 2005 Massachusetts Institute of Technology

[6] Prof Narendra Kumar, Dean DIT School of Engineering and a student of PhD at Manav Bharti University, Solan, Himachal Pradesh (India)," Data Centric Storage in Wireless Sensor Networks"in International Journal of Advances in Engineering Research (IJAER) 2011, Vol. No. 1, Issue No. VI, JUNE

[7] Olaf Landsiedel, Katharina Anna Lehmann, Klaus Wehrle Wilhelm-Schickard-Institute for Computer Science University of Tubingen, Germany "T-DHT: Topology-Based Distributed Hash Tables"

[8] Karp, B. and Kung, H.T., Greedy Perimeter Stateless Routing for Wireless Networks, in Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000), Boston, MA, August, 2000, pp. 243-254.

[9] Tutorial for the Network Simulator ns-2 "http://www.isi.edu/nsnam/ns/tutorial/"